Csdn
全面关注软件人与企业成长
Programmer 2005年 8月刊
程序员



IBM软件十年之旅

以硬件著称的IBM公司，不但将触角伸入软件世界，而且已经对全球软件格局产生着越来越大的影响与带动作用，在IBM软件集团成立十周年之际，软件产业的IBM现象，已成为一个不得不说的话题。

P18 为了拯救一个程序员的生命
P36 怎样做实用主义程序员
P58 网站项目成功管理实践

技术专题
P76 C#编码规范

随刊赠送 Sun Solaris 10 DVD 光盘

www.csdn.net
ISSN 1672-3252
9 771008 225023 15>
邮发代号:2-665 定价:10元

程序员 Programmer
2005年 8月刊
全面关注软件人与企业成长

Programmer 程序员 www.csdn.net

2005年08月1日 第8期 总第114期

主管：中国社会科学院
主办：中国社会科学院文献信息中心
协办：北京百联美达美数码科技有限公司
网址：http://www.csdn.net
国际刊号：ISSN 1672-3252
国内刊号：CN11-5038/G2
邮发代号：2-665
广告经营许可证号：京东工商广字 0188 号
出版：《程序员》杂志社

总编：黄长著
社长 / 常务副总编：张悦校
副社长：蒋 涛
编委会：黄长著 张悦校 陈洋彬
蒋 涛 唐 琦 曾登高
叶 展 许 伟 叶 丁
执行主编：唐 琦
副主编：闫 辉

编辑部主任：孟迎霞
技术主编：孟 岩
责任编辑：欧阳瑾 熊建国 霍泰稳
崔 轶 赵建凯
美术总监：张浩祥
美术编辑：吴志民 牛新疆
电话：010-64351458
投稿信箱：editor@csdn.net

发行部：010-64351456
信箱：sales@csdn.net

广告部：010-84540265/64348410
信箱：adv@csdn.net market@csdn.net

上海联络处：孙峻峰
电话：021-63619663-801
传真：021-63619457
信箱：shoffice@csdn.net

西南联络处：谭 皓
电话：023-63810152
信箱：tanhao@csdn.net

台北联络处：上准资讯股份有限公司
划拨帐号：18459461
户名：吴权昌
电话：02-29262357
信箱：henry@csdn.net

读者服务部
网上订购：www.dearbook.com
读者信箱：reader@csdn.net
地址：北京市朝阳区酒仙桥路 14 号兆维工业园 A2 区 1 门 4 层
邮政编码：1000016
电话：010-64351425
传真：010-64348545

法律顾问：正见永申律师事务所

印刷：北京乾洋印刷有限公司

出版日期：每月 1 日
零售价：RMB 10.00 元 新台币 260 元
HK$25.00（港、澳）
US$6.00（海外）

本刊文章版权所有，未经许可不得转载。

发现装订错误或缺页，请将杂志寄回本刊读者服务部，即可得到调换。

人物&报道

名人堂

8 Theo de Raadt：大嘴 OpenBSD 创始人

IT界从来不缺个性张扬的开发高手。Theo De Raadt就是其中的代表，他不仅负气出走 NetBSD而重新打造出 OpenBSD，更因为“嘴臭”而闻名。

10 程序天下事

14 封面书摘 & 网站动态

16 网站精彩回眸

感悟

18 为了拯救一个程序员的生命

特别策划

20 IBM 软件十年之旅

- IBM 软件技术十年评述
- 大象起舞，蚂蚁离开
- 我眼中的 IBM

看当前软件业界，除微软、SUN、BEA 等企业之外，以硬件著称的 IBM 公司，不但触角早已深入软件世界，而且对全球软件格局产生着越来越大的影响与推动作用，在 IBM 软件部门成立十年之际，分析 IBM 现象，已成为一个不得不说的话题。
为了更好地剖析 IBM 软件十年的发展历程，挖掘技术发展的线索，追踪其趋势变化，本刊特别组织了这期专题策划，剖析 IBM 软件十年技术策略的背景，产品的发展路线，以及第三方对 IBM 软件发展的观点。

人物专访

36 怎样做实用主义程序员

Dave Thomas和他的合作伙伴Andy Hunt一起开办了一家咨询服务公司，专门帮助团队更好地编写软件和提高软件质量。本文是对他的专访。

40 程序其实也是一种艺术
——袁红岗的程序人生

许多人认为程序员每天面对的是枯燥的工作，但对于金蝶中间件技术总监袁红岗来说，写程序其实也是一种艺术创作，需要想像力、激情和思考。

公司

42 开放的 SAP 拥抱开发者

SAP 作为全球企业管理软件的王者，最近从原有的封闭技术体系中走了出来，推出了新的开放性的技术平台。

软件创业

44 一个行业软件公司的成长

一家从事进出口贸易相关领域软件开发的行业软件公司是如何寻找到市场的商机，又是如何利用技术产品和项目运作完美结合的呢？他们做行业软件开发的经验与您共享。

46 声音 & 幽默

管 理

管理专题

47 软件项目的管理现状和出路

- 走出软件项目的死亡之旅
- 软件项目成功的要素

尽管采用了系统化的软件项目管理方法和质量保证体系，但仍有许多软件开发组织的软件项目成功率保持着“死亡”状态，专家为您解读这其中的问题所在，并为项目管理指明出路。

项目实战

54 来自死亡项目的验尸报告

“验尸报告”在软件项目管理界已经成为一个热点名词，本文从一个测试人员的角度，为项目的失败做出了经验性总结。

58 网站项目成功管理实践（上）

其实网站开发和产品的开发并没有什么不同：如何把握项目进度？如何掌握项目团队？作者刘振飞通过产品研发管理经验亲手实践一个互联网项目的全程。

过程改进

63 调查：方兴未艾的管理咨询
——CMM/CMMI 在中国

面对方兴未艾的 CMM/CMMI 评估认证热潮，常常让人想起 ISO 在中国的历程，“形式主义”是否仍然在重现？通过本刊调查，您将重新认识管理咨询热潮。

66 Microsoft为何不能按时交付产品

Microsoft产品发布的推迟早已经是家常便饭，但是您可能从未听说过有人真正回答这个软



件巨头为何一次又一次的拖延交付日期，且看分析师找到的答案。

测试员专栏

68 测试过程的监控方法

和任何工作一样，测试也需要有一套完善的监控方法。尽管这看起来有些教条，但是作为一个优秀的测试人员，这些工作是必须牢牢掌握的。

软件工程

73 鱼与熊掌

——完美主义架构师的梦魇

尽管我们都有将软件设计得更加完美的美好愿望，但这几乎不可能的事实让软件架构师们辗转反侧。折中成了优秀架构师的必要选择。

技 术

技术专题

76 C# 编码标准

在连续了几期对技术趋势的分析、解决方案的回顾和开发框架的评析之后，我们先后收到了一些读者对于技术专题的反馈。在肯定了前面内容的同时，也希望能够介绍一些面向一线开发人员的"案头手册"。针对这个问题，这期《程序员》的技术专题为C#开发人员准备了一份"行为规范"，希望该规范可以使不同阶层的C#开发人员都能够在时间、效率以及质量上有所收益。

安全视点

91 恶意代码的亲密接触

——文件搜索和API寻址

在上期介绍了病毒的相关基础知识后，从这期开始我们就要深入病毒内部，这一期我们会看到API函数寻址以及文件搜索的技术，为后面更加深入的探讨打下良好的基础。如果你已经准备好了，就让作者带你进入这无所不用其极的病毒技术世界吧。

97 杀毒软件探秘

——静态杀毒软件的构建

反病毒技术在很多朋友眼中总有一些神秘感。在接下来的几期文章中，我们会邀请Rising的技术专家向大家介绍关于杀毒软件所用到的方方面面的技术，以求使大家更好的明白杀毒软件到底是如何工作的，这一期，我们将从一个真实杀毒引擎的原型开始。

企业开发

101 基于模式的UI设计实战

在上一期文章中作者留下了一个悬念：什么样的GoF模式适合架构UI框架中的三种模型呢？作者通过对环境操控模型、导航逻辑模型等方面给出了答案。

105 基于Web方式的CMS设计与实现

作者在2004年的年底，开发一种与平台无关、结构良好、拓



展性强、可以解决实际问题的 CMS 系统。现在回过头来总结系统的整个设计和开发过程，作者愿意与各位读者交流一下其中的经验和体会。

Java

110 趣谈 Java 语言十年

开源之窗

114 Eclipse RCP 入门

Eclipse的插件机制更是提供了一种令人耳目一新的开发方式。现在国内做 Eclipse 插件的渐渐多了，并且很多用 SWT/JFace和插件方式来开发应用项目中的界面。本文作者通过一个开发实例，以 Eclipse3.0 为基础，介绍了一些 RCP 所特有的东西。

软件整合

119 现代软件 GUI 分析与 GUI 柔性设计

Win 95 推出后的十年，软件开发工具日渐成熟了，在日益华丽的界面的背后，各大技术强悍的公司依旧在寻求它们的游戏规则，使得软件的游戏规则依然因他们而动。

产品&应用

产品测试

124 用友政务软件性能测试与调优

软件性能是衡量软件质量优劣的一个重要指标。性能测试是为描述测试对象与性能相关的特征并对其进行评价而实施和执行的一类测试。

书评

126 Jbuilder 2005 的百科全书

《精通 JBuilder 2005》不但全面深入地介绍了 JBuilder 工具本身，而且包含了很多技术知识和经验。

新书节选

128 现实中的软件工程

理想状况下，“软件工程=过程+方法+工具”。然而，工程成功的真正关键并不在于把团队“组织”得有多好。即使在团队中都显示出有条不紊，一样会面临失败。

131 Amazon.com 开发技术图书每月观察

132 新书上架

Maglog

133 走进 JCP

黄柳青专栏

134 打造面向构件的企业级应用

Theo de Raadt

大嘴 OpenBSD 创始人

文/方茜

IT界从来不缺个性张扬的开发高手。Theo De Raadt，这位名字古怪的主儿不仅负气出走NetBSD而重新打造出OpenBSD，更因为“嘴臭”而闻名：因为痛斥美国对伊战争，引来美国国防部高级研究计划总署（DARPA）撤销了对OpenBSD项目的支持补助金；同时还因为抨击Linux，在开源社区掀起旷日持久的口水大战。

出生在南非的De Raadt九岁时随他的父母移民到加拿大卡尔加里市，并在那里完成了计算机本科学习。1993年，De Raadt与三位朋友合作开发了NetBSD操作系统。年轻气盛的他总挂在嘴边的一句话就是：“闭嘴，好好地‘黑’它一把！”开源社区里的De Raadt也经常这样出言不逊。最有名的事件是，有位程序员跟他发生口角，“他（De Raadt）完全是一个输不起的人。我告诉他别说了，做事吧。”稍后这位程序员发Email向他道歉，可是De Raadt毫不领情，回了一封尖酸刻薄的信，更恶心的是把这封信转发给数以百计的程序员以供消遣。结果，人神共愤的De Raadt被踢出了由他创建还不到一年的NetBSD开源组织。

一气之下，De Raadt另起炉灶，这就是OpenBSD。1995年，OpenBSD第一版发布。作为一款类Unix的开源操作系统，OpenBSD强调的是它的正确性、标准化与可移植性，更把安全作为关注的焦点，由





此催生出了诸如OpenSSH这样的工具箱。现在，OpenSSH已成为很多Unix和Linux发行版安全在线通信的事实标准。另外，这一努力的其它副产品还包括可靠的BGP实现、IPSec协议栈和数据包过滤器。十年过去了，De Raadt得意地说：关心OpenBSD远远多于NetBSD。

别以为De Raadt吸取了在NetBSD社区的教训。2001年，美国国防部先进研究计划总署（DARPA）资助OpenBSD二百三十万美元，用于增强该软件的防暴安全性。可是，大嘴依然的De Raadt在接受某加拿大报纸采访时，强烈谴责美国政府的对伊战争，说这是为了石油的贪婪把戏。很快，DARPA就中止对该项目的资助，声称这是因为世界性事件与越来越多民族国家构成的威胁增多的缘故。De Raadt并不感到沮丧，他说即使没有DARPA的支持，OpenBSD团队还是可以继续运作。他表示，从他发出邮件说明补助金遭撤销的情况后，已经陆续收到许多捐款。他认为可以由捐款支持运作，或尝试取得其它补助。

此番风波后，De Raadt还是不改大嘴本性。在接受《财富》杂志采访时，他又把火烧到Linux身上。“仔细瞅瞅Linux，它简直是奔着下一个微软去的。”它是一个低质量的软件。他们的东西一点也不好。”（Linux）系统中有许多部分只是一些寒碜的小玩意儿。它碰巧能运行起来。”但对自己的软件，De Raadt可一点不吝惜赞扬：“我认为我们的代码质量更高，因为这是我们关注的重点。”“人们使用它（OpenBSD）但不宣扬它。它真是悄无声息。”尽管，迄今为止OpenBSD已被下载数百万次。对此，Linux的创始人Linus Torvalds并没有发表什么过激的言论，他只淡淡地说了句de Raadt这人有点“难相处”。

而当被问到OpenBSD使用起来非常难、非常痛苦时，De Raadt毫无歉意地把这个问题扔给别人：“我们更多的时间是花在让我们的东西更好，而不是让它符合大众的口味。”

尽管De Raadt个性狂放、树敌颇多，但开源组织FSF（自由软件基金）还是在2004年颁发给他自由软件贡献大奖，足见OpenBSD在开源运动中的重要地位。

微软技术



主持人：Kaneboy
涂曙光，微软（中国）有限公司产品技术专家，博客堂成员。专注于.NET开发，从事Office System、SharePoint等产品相关的技术支持。

这几个月以来，Ajax（Asynchronous JavaScript and XML）俨然不断的漂浮在Web系统开发人员的眼前，并极大的冲击着传统的“提交——回发——重绘”式的Web系统基本运行结构。Ajax以无刷新、依靠客户端直接与服务器交互等技术，带给了用户比以往好得多界面交互感受。Google的Gmail就是一个把Ajax用到极致的例子，当用户使用Gmail时，会感到它比普通的Web Mail系统反应速度更快、交互性更好。

其实Ajax所用到的关键技术，包括DHTML、JavaScript、XMLHTTP等，很早就已经被大部分浏览器所支持了，所以与其说Ajax是一项全新的Web开发技术，不如说Ajax是对以往客户端各种Web技术的一种整合、提炼和总结。

Ajax.NET是网上非常有名的一个.NET平台上的Ajax项目，作者提供了一个非常好用的类库，极大的降低了在ASP.NET中使用Ajax技术的难度。最近，作者已经决定将Ajax.NET开源，并已经在SourceForge上建立了项目站点。读者可以通过http://sourceforge.net/projects/ajaxnet-library来访问相关的信息。

除了网络上第三方制作的各种Ajax类库、帮助工具之外，其实微软已经启动了一个基于ASP.NET 2.0的Ajax项目，名字叫做“Atlas”。Atlas的目标是让Web开发人员不需要具备太多的客户端脚本知识，就能够轻松地创建基于Ajax技术的Web程序，并能够编写更加快速、调试更方便的类Ajax程序。实际上在ASP.NET 2.0中，微软已经在ASP.NET架构中加入了异步客户端脚本调用功能，使构建Ajax类型的Web程序更加方便，但是Atlas将走得更远。

Atlas客户端脚本框架（Atlas Client Script Framework）是一个丰富的、100%面向对象的、扩展性极好的JavaScript框架，即使不是一个JavaScript、XMLHTTP方面的行家，开发人员也能使用这个框架开发出包含大量的客户端脚本、DHTML和XMLHTTP的具有丰富界面的Web程序。

微软还会提供一组新的基于Atlas的ASP.NET服务器控件，使这些控件充分使用Atlas所带来的好处，具有更好的用户交互性。任何Ajax系统都需要有某种途径能访问服务器端的代码和操作，Atlas使用了最通用的Web Service。通过与Web Service的无缝集成，任何使用Atlas框架的Web应用程序都能直接访问任何ASP.NET ASMX Web Service和未来的Indigo Service，Atlas会自动负责创建Proxy对象，并对对象进行序列化等操作。由于拥有Atlas这种与Web Service无缝集成的能力，重要的服务器端的业务逻辑代码只需要编写一次，并通过Web Service发布出来，就不但能提供给任何常见的Windows应用程序、Web应用程序，还能直接提供给Web客户端程序，实现了资源的最大化利用。

ASP.NET 2.0提供了开发人员大量现成的功能模块，比如管理用户信息的Profile模块、管理用户和角色的Roles模块等，这些模块将也能被Atlas直接使用，Atlas框架将能够在Web客户端，直接访问到这些功能模块，获取其中的数据信息。

Atlas在今年9月份的PDC（专业开发者大会）上将发布一个开发者预览版，并随即建立一个专门的站点，用来发布新的版本、接受反馈、提供示范等。

可以预见，通过.NET社区和微软的共同努力，Ajax在ASP.NET世界中将很快的找到自己的位置，并在Web开发领域拥有自己的一席之地。



● 6月30日“2005微软开放日”在北京举行。此次活动以“激发企业潜能——新一代IT架构和全新工作方式推动企业高效成长”为主题，内容围绕“安全、互联、高效”三大核心思想展开，主要包括：新一代IT架构如何优化企业运营管理，保持良好的竞争力；通过成功案例，诠释如何建立安全、互联、高效的信息化平台；全新工作方式如何提升营运效率，引领企业飞速发展。此外，电子政务以及教育信息化论坛，也成为本次微软开放日的亮点之一。

● 7月12日，游戏学院与Autodesk公司在长城饭店联合召开新闻发布会，宣布双方开展战略合作。来自中国软件行业协会游戏软件分会的副会长刘金华、国家863网游负责人、游戏厂商、双方企业的负责人和众多媒体参与了此会。这次双方合作的主要内容是：Autodesk公司授权游戏学院培训中心为“Autodesk授权游戏培训中心”，游戏学院的学员从此又可多获得一项高含金量的认证——“Autodesk游戏动画设计师认证”。

● 7月13日，趋势科技宣布，鉴于间谍软

Java技术



主持人:透明
CSDN 技术主编。

流火的七月,围绕着开源的话题,温度也不亚于室外的气温。JBoss的Marc Fleury又一次冒出来大放厥词——当然了,可爱的Marc总是在大放厥词。话题嘛,自然还是离不开他的"职业开源"。虽然Fleury的这套说辞气跑了Rickard Oberg,惹得Hani Suleiman三天两头拿他开涮,但公正地说,JBoss对于开源世界的贡献是巨大的:是他们率先在J2EE开源社群把"我们要赚钱"的口号大张旗鼓地喊出来,并且开拓了一些值得借鉴的商业模式。

在这个技术动荡变迁的关头上,Java世界的开源又迎来了一波新的高潮。按照BEA发布的调查报告,全球前2000家公司中有70%以上在使用一种或多种开源框架,多达28%的公司在开发环境中使用了一种以上的应用服务器。作为对潮流的顺应,BEA宣布了支持开源的一揽子计划,其中包括在Workshop IDE中内嵌对Spring框架的支持——别忘了,WebLogic Workshop本身也将改弦更张,转到Eclipse平台上。

而Sun公司本身对待开源的态度却没有像BEA这么积极。虽然Sun已经在"玻璃鱼"(Glassfish)项目下开源Java System Application Server 9.0(JavaEE 5规范的实现品),但采用CDDL授权协议一直引来诸多诟病。本月接受InfoWorld的采访时,Sun公司负责Java Web Service和工具市场的副总裁Joe Keller又透露说:Sun公司不打算加入Eclipse组织,仍然会继续发展自己的IDE项目(也就是NetBeans)。虽说有竞争是好事,但拥有两个基于插件体系结构的IDE项目就意味着大量的插件需要重复开发。譬如一直被寄予厚望的JSF,Exadel已经为它开发基于Eclipse的插件JSF Studio,Sun又不辞辛劳地做了一套基于NetBeans的插件。J2EE web应用开发的便利程度一直赶不上ASP.NET,和IDE的长期分裂不无关系。

众多工具/产品纷纷开放源码,连BEA和SUN这样的厂商也接连把自己的产品开放出来,IBM在背后的推动是一大原因。本月IBM alphaWorks又向MIT等高校开放了一批技术源码,另外别忘了,Eclipse和Apache Jakarta的背后都有IBM的大力资助。CSDN进行的一次调查显示,近80%的中国开发者首选的免费servlet服务器是Tomcat,Resin才占到16%。在这种强大的压力之下,想从开发工具上收费确实难度不小,所以开源也成了诸多厂商不得已的下策。

不过在我看来,开源热情的空前高涨似乎表明Java技术已经发展到一个相当成熟的阶段——诞生10年的Java如果再不成熟那也奇怪了。在今年的JavaOne大会上,Sun公司已经宣布取消"J2XE"的命名法,而且JavaSE/EE 5在技术上的跨度非常之大,很大程度上应该将它们看作一种新技术的起步,而不仅仅是J2SE/J2EE的延续。面对近乎全新的技术架构,洗牌重整是不可避免的,大范围的开源热潮应该只是一个前奏。

在撰写这篇短文的过程中,我发现自己不断地在翻阅CSDN Java频道(Java.CSDN.net)的历史新闻列表。本月Java频道又进行了一次改版,自从完成改版之后,Java频道的信息量和组织形式都有了很大的改进,相信对于读者会更有帮助。此外我们还组织了一系列的有奖征文活动,希望可以给读者在炎炎夏日中带来一点乐趣——获奖文章的作者将赢得今年JavaChina大会的门票,所以请赶快参加吧。

件威胁日益严重,间谍软件已经成为2005年企业IT投资的前三大优先考虑事项之一;IDC统计,目前67%的计算机中都存在间谍软件;间谍软件可导致关键业务系统速度降低、支持成本增加以及生产效率下降等各种问题……趋势科技推出了以防间谍软件为主打功能的新方案——趋势科技防毒墙网络版OfficeScan 7.0。它与趋势科技的明星产品InterScan 网关网络安全套件相配合,可以帮助企业用户实现从网关到客户端/服务器,全程免受间谍软件的骚扰。

● 7月17日,10名来自微软亚洲研究院的实习生登上了飞往美国西雅图的班机,开始了他们憧憬的数字梦想之旅。他们是应微软公司邀请,参加一年一度在比尔·盖茨家中举行的微软实习生聚会。这也是微软公司第一次邀请来自美国本土机构以外的实习生,微软亚洲研究院实习生访问团也是唯一的国外受邀代表参加这一活动。

● 7月19日全球领先的接入架构解决方案提供商——思杰系统有限公司宣布在中国正式推出Citrix Access Suite 4.0升级版和最新产品Access Gateway,思杰公司将与合作伙伴合力打造信息接入新时代,帮助企业实现业务的灵活性、员工的流动性、并提升安全性能以及企业应用和资源的可控性,为企业节省成本开支,进一步提升中国企业核心竞争力。

软件工程与项目管理



主持人：潘加宇
UMLChina 首席专家，潜心研究和实践UML/UP相关技术的应用。

又有消息说Visual Studio 2005 Team System的正式发布也许要等到2006了。微软要靠Team System深入到应用生命周期管理这个领域中，所以要谨慎行事。三年前，这个领域的主要伙伴Rational被竞争对手IBM收购以后，微软就要自己走一条非UML的道路。

这时，我们看一下Borland公司的举动，它力图架起微软方案和UML方案之间的桥梁。虽然Rational仍然表示将会继续支持.NET，但Borland的Together产品主管认为，Rational还没有在.NET上做太多工作，似乎也并不想支持以后的平台，因此Borland要进一步在.NET上帮助开发人员简化在UML建模上的工作，最近更新的Together 2005 for Visual Studio .NET为Visual Studio .NET提供了UML2.0的支持。

Together能用来干什么，只是用来建模然后生成代码？Borland的传道者李维最近的讲座"站在巨人的肩膀上掌握软件财富"通过用Together剖析几个著名的开源框架，很新颖地提出：它还能用来帮助开发人员学习开源代码的架构和代码精华。

微软在VSTS中没有提供需求管理这部分功能，实际上将由Borland来提供。6月的微软TechEd大会上，Borland宣布了这一点。Borland的需求管理工具是CalibreRM，它的Team System版本预计将在2006年初，在VSTS之后推出。

同样在TechEd 2005上，致力于自动化过程建模和剪裁的公司Osellus演示了IRIS - on - Demand for MSF，用于剪裁MSF for Agile和MSF for CMMI，通过它，团队可以以很少的成本对MSF进行剪裁－发布、剪裁－部署。Osellus也同样提供针对IBM的Rational Unified Process的剪裁工具，通过RUP桥接（RUP - Bridge）技术，把RUP的所有内容导入到IRIS，保存到符合SPEM（Software Process Engineering Metamodel）的仓库之中，在此基础上进行剪裁。

最近的另一件值得关注的事件是Telelogic收购了Popkin。Popkin是最早出现的CASE工具开发商中独立存活下来的最后一家了。这两家的产品中，Telelogic的TAU和Popkin的System Architect都是基于UML的工具，但Popkin更注重于业务，而Telelogic倾向于软件开发。Telelogic这次收购显示出它对业务建模的重视，不知是否和最近的"模型驱动"的趋势有关？

嵌入式移动开发

主持人：马宁
微软最有价值专家，Windows Mobile开发者。



火热的七月已经来到我们的身边，本月来自微软的消息仍旧占据头条。7月12日，微软在日本横滨发布了Windows Automotive 5.0。Windows Automotive 5.0是用于车载电子系统的软件平台，和Windows Mobile 5.0一样，该产品也是基于Windows CE 5.0操作系统。和Symbian只提供给智能手机不同，Windows CE应用于从个人电子消费品到工业设备的广泛的领域之中。Windows Automotive 5.0提供了3D导航软件和丰富的娱乐功能——支持Windows Media、MP3和DVD播放。

微软将在今年秋天提供一个新的微软认证——移动解决方案认证（Mobility Solutions Competency），这样认证主要考察程序员开发和发布Windows Mobile应用程序的能力，另外还包括安全和系统管理方面的考察。

7月9日，微软发布Microsoft Global Contact Access。该软件允许你的Windows Mobile设备无线访问Exchange服务器，查看你同伴的工作进度，看他们是否有空闲，还可以给多人发送会议请求。如果你需要远离办公室进行工作，该软件将是一个很好的帮手。

本月Palm也发生了很大的震荡。7

在开源世界里，总是时不时蹦出一些有趣的项目来给你一点小小的惊喜。虽然它们中间的大多数很快就会淡出人们的视野。但是如果能够带给我们一些帮助，带来一些灵感，那也就不枉走一回了。

前几天无意间发现的一个项目——Phalanger，中文意为“袋貂”，一种产自澳大利亚或新西兰的小型有袋类动物；由一群捷克的开发者维护的一个开源项目，目的是为了将PHP5移植到.NET平台上。Phalanger这个项目并不出名，但令人惊喜的是它目前的移植度已相当高。据其主页上介绍，在一个Phalanger要求的环境下：IIS＋ASP.NET＋.NET Framework 1.1＋Phalanger，可以在修改极少的甚至是不修改的情况下运行PhpMyadmin和phpBB这样典型PHP应用。

众所周知，PHP一个吸引人的地方在于拥有一套庞大扩展模块。这些模块以函数库的方式提供给开发者。对于许多初学者而言，不需要理解面向对象的许多复杂语意，就能迅速利用PHP开发出可以运行的程序。因此要想完整地将PHP移植到.NET这样的虚拟机平台上，仅仅实现一个PHP语法分析和编译器是不够的。Phalanger小组在移植函数库的方面做了不少工作，相当部分已经用C#语言在.NET上进行了重写。当然了，还不止这些，Phalanger还能够调用.NET平台上其它语言所写的库。并拥有一个VS.net 2003插件，帮助开发者在VS.net环境下编写PHP代码。另外Phalanger的性能还相当不错，由于在.NET上能够被编译，Phalanger执行效率要高于原始的PHP执行环境。以执行phpBB为例，Phalanger的性能大致是原来PHP环境的2.5倍，是原始PHP+Zned Optimizer的1.7倍，性能提升颇为可观。

目前Phalanger的版本为1.0rc1，并兼容于PHP5.10，离正式发布的距离并不遥远。当然，或许有人会质疑Phalanger到底会有多大的用处？确实，Apache和PHP解释器都已被完美的移植到了Windows平台上。并没有非常充足的理由使用Phalanger来开发和部署新的PHP应用。但是当你主要工作于ASP.NET环境时，而需要运行诸如phpmyadmin/phpBB这样系统，那么Phalanger是非常合适的。

Phalanger其实只是现在风起云涌的动态（脚本）语言向虚拟机（.NET/JVM）平台上移植大军中的一员。从 http://www.dotnetpowered.com/languages.aspx 上可以看出，在.NET平台上，几乎所有主流语言都已经或正在移植到上面，不仅有Python、Perl、PHP、Ruby现在非常热门的语言，也有Ada、Cobol、Fortran、Prolog、Scheme这样的传统主流语言。这也从一个角度表明了当初微软为.NET平台制定的跨语言战略的初步成功。虽然到目前为止，我们依然无法看出这样一个“万能的”跨语言平台究竟能为我们的开发效率上提升带来多大好处。

不光.NET平台，Java平台也是如此。随着人们对动态语言兴趣越来越浓，Java平台上也迅速诞生了一批诸如Jython/Groovy这样的动态语言。

两个平台出现类似的情况，也许并非偶然。这意味着现在开发者所形成的一个基本的共识：静态语言和动态语言都拥有对方所无可替代优势，这就如同一个硬币的两面。当他们统一在一个平台下，更容易彼此优势互补。静态语言拥有严密的编译期检查、更好执行效率，可以形成应用的框架；动态语言无需编译，更加灵活，可以完成更加复杂的业务逻辑。二者的有效配合也许将是今后一段时间里最有效的软件开发模式。



主持人：汤韬
开源爱好者。

Open Source

月14日，PalmOne正式更名为Palm，在经历了PalmOne与PalmSource的分分合合之后，Palm品牌终于回归硬件设备制造商PalmOne。在PalmSource宣布致力于Linux开发后，Palm将不再和操作系统紧密结合，以后我们很可能在市场上看到基于Windows Mobile操作系统的Palm设备。曾几何时，在PDA市场上叱咤风云的Palm竟然到了今天的地步，真是沧海桑田。

近日ICANN宣布.mobi顶级域名将提供给移动设备内容供应商使用。该域名由主要的移动手机制造商和操作系统提供商资助，其中包括微软和GSM协会。拥有该域名的网站将为移动设备提供更加快捷的网络浏览服务。

似乎是为了和ICANN的消息呼应，Yahoo于7月8日推出了自己的SMS搜索服务。Yahoo铁了心要做Google的追随者，这次他们的SMS搜索服务和去年推出的Google SMS服务十分类似。用户可以通过该服务来搜索字典解释、邮政编码等信息。不过目前这项服务仅限于美国。

本月最后的一条消息似乎和技术无关。在伦敦爆炸发生后，英国已经展开了一项名为ICE的活动（In case of Emergency，在紧急情况下）。这个活动建议大家将你在遇到紧急情况时需要联系的电话号码，用ICE的名字保存在手机联系人中，这样方便医生用最快的时间和你的家人取得联系。这个看似简单的举动也许会在紧急情况下帮我们和救援人员的大忙，不知道中国什么时候才可以展开这项活动。

ICE计划提醒我们，技术的最终目的是为了让全世界的人们过得更加幸福。

MSDN Magazine



关于多线程应用程序，每个开发者都需要知道的

十年前，只有那些最猛的系统程序员才会在面临开发多线程程序时担心其复杂性。大多数程序员为了避免那些问题，坚持写单线程顺序执行的程序。然而现在，带有多CPU的机器已经很常见，很快，不会开发多线程应用程序的程序员会面临不利局面，因为他们将无法充分利用强大的设备计算能力。

不幸的是，编写正确的多线程应用程序是不容易的。程序员们通常难以理解为什么其他的线程会先于当前线程访问内存。更坏的是，当出现一个错误时，程序大多数时候还能继续运行。除非资源紧迫，否则错误很难被发现，并且错误信息也很少，不便于调试。其间总结了单线程和多线程程序的主要区别，如你所见，让多线程程序一次性运行正确需要花大力气。

Dr. Dobb's Journal



如今的软件开发环境——你知道，就是你整晚盯着的屏幕——已经远远超过以前的IDE。它们易于配置，功能强大，支持插件。本月的DDJ我们关注几种新生代的开发平台，包括Eclipse 3.1，NetBeans 4.1，Visual Studio 2005等等。其他内容还有对.NET 2.0的详细介绍，Linux线程模型的新实现方式，到64位系统的转换工具和编译器。我们还将展示一个可复用的Duff's Device（听不懂？读完杂志就知道了）。

Visual Studio Magazine



在Yukon中加密和解密数据

SQL Server 2005的引擎级数据加密功能可以保护机密信息不被非法访问，此外还提供了原生的密钥管理功能。

使用观察器（Visualizer）简化编程

充分利用Visual Studio 2005中带的观察器来提高开发效率，并学习如何创建自定义的观察器。

正确的构建服务型组件

利用本文提供的实用指南来正确地创建服务型组件，减轻写代码和维护的麻烦，为现在——也为以后。

测定性能需求

学习如何利用Windows管理接口（WMI）来正确测定应用程序的性能需求。

JAVAPro



以质量向企业进军

企业开始寻求更加省钱的中间件以取代昂贵的商业产品，开源软件因此找到了加入企业商务活动的路子。Java Pro本期将独家专访JBoss公司的创始人、主席兼CEO：Marc Fleury，与他探讨JBoss进军企业级中间件市场的途径，开源软件的风行潮流、授权协议、质量保障等等。

Eclipse新版发布

IBM开放了Eclipse的源码仅1年多时间，人们对这个开发平台的兴趣和认可已经显著增长。从本文中可以了解更多Eclipse的发展和成功案例，以及它如何为Java社群提供一种免费的高适应性和可扩展性的开发平台。

架构最重要

尽管很多的IDE和应用程序框架使得编程更加容易，但它们对于系统架构的设计没什么帮助。本文将说明为什么软件架构的设计应当是开发者首要考虑的因素。

网站动态

EARTHWEBNEWS.COM

http://news.earthweb.com

HP计划在全球裁掉约10%的员工，据说这可以为HP节省约19亿美元。HP计划在将来的6个季度里共裁掉14500人。通过工作的削减和重组，HP可以节约16亿美元的员工工资和3000万美元的额外开销。“在深刻分析了我们的商业状况之后，我们计算出了一个可以发挥HP最大潜力的公式，”HP CEO Mark Hurd在一次声明中说：“无论是对我们的客户、合作伙伴、员工还是股东我们都可以做的更好，并且我们也愿意做的更好。”据HP表示，这次裁员的重点是在支持部门上，例如IT，人力资源和财务等部分。而剩下的裁员工作将会分布到不同的商业单元上。为了推动这次裁员的顺利进行，HP还会对在美国工作的元老级员工提供一定的退休补偿。

Slashdot
News for Nerds. Stuff that matters.

http://slashdot.org

近日，一位匿名的读者在slashdot上写了一篇名为“现在，Linux的发行版是不是太多了？”的文章。该文章从多方面指出了Linux发行版百花齐放的现象带来的影响并指出：现在是Linux该团结起来的时候了。这篇文章虽然只是把谈论的焦点放在了Linux桌面应用上，但是其中所谈及的一些观点却影响了整个Linux社区。例如，文中明确指出了“Linux应该是Windows的强有力竞争者。”另外，文中还提到：“Linux至今没有超越Windows的一个鲜为人知的原因是没有解决好如何有效的分配和利用资源而已。众多的发行版带来的后果是Linux社区过度分散，以至于所有特定的技术都只能在局部发光，很难聚在一个点上。”而面对这个问题，读者表示了不同的看法。支持者的态度：“我们应该成立一个委员会来负责团结的工作”，当然也有反对者说：“这是一个让Linux破产的绝佳工作。”

TheServerSide.COM
Your Enterprise Java Community

http://www.theserverside.com

本月的TSS显得有些沉寂，不知道是不是因为刚参加完JavaOne，“大拿”们还没有从兴奋与疲劳中缓过劲来。比较有趣的是两个技术对比。首先是Aaron Rustad对J2EE与Ruby on Rails进行的对比。Rails是一个完整的体系，从表现层的MVC到持久层的O/R映射一应俱全，所以要对比的目标也得是流行的J2EE整体架构——通常会包含Spring与Hibernate。Rustad指出，在一些核心架构模式上，Rails和J2EE的思路可谓大同小异；但在对“企业应用开发”这件事情的理解上，两者有明显的差异，体现为Rails更倾向于编码而J2EE更倾向于配置文件、Rails更倾向于动态化而J2EE更倾向于静态化。

另一个有趣的对比是在JUnit、JTiger和TestNG这几种单元测试框架之间进行的。JUnit自然是毫无疑问的事实标准，而另外两位参赛选手也有各自的长处，值得称道的是TestNG可以接续上次出错的地方开始运行，以及JTiger正确/错误报告突出的条理性。作者给出的建议是：在做比较重要的项目时，挑选多款测试框架共同使用，综合各个软件的长处，扬长避短。

BEA近日宣称他们发布了最新版高端事务处理器Tuxedo 9.0。Tuxedo被整合在BEA的AquaLogic平台上，并对C、C++、COBOL提供事务管理。BEA对Tuxedo进行了如下说明：“BEA的Tuxedo系统和Service Request Broker (SRB)非常相似，它提供面向服务的基础架构，从而可以跨BEA Tuxedo 流程和应用，高效地路由、调度、管理各种请求、事件和应用队列。”Weblogic Integration将通过WebLogic Tuxedo连接器提供服务，WebLogic Tuxedo支持在BEA Tuxedo的C、C++、COBOL应用程序、BEA WebLogic Server、J2EE平台工具和应用程序之间的完整事务和安全传导，但是，目前唯一的不足是仍不支持代理事务。这势必会在Tuxedo和WebLogic之间带来一些问题。

artima developer
A community of programmers who care about their craft

http://www.Artima.com

Bruce Eckel连续在blog里写了几篇评价Java基础语言构造的文章：首先是一篇“泛型是有害的”，然后是一篇“作为参数的Collection被滥用了”——这第二篇文章相当有趣，他认为很多时候人们应该传递的参数是Iterator，而不是Collection（甚至Collection的子类例如List）。随后，他又补上了一篇“Collection vs. Iterator”，详细分析两者之间的语意差别。

Frank Sommers的一篇“用JDM API挖掘数据”介绍了Java数据挖掘API（JDM）的用法。数据挖掘也算是企业应用的一类典型业务了。JDM API先后制订了两个JSR规范，分别是JSR-73（JDM 1.0）和JSR-247（JDM 2.0），其目标是为开发者提供一个“如同JDBC一般好用”的数据挖掘统一接口。至于这个承诺是否能够兑现，就请看Sommers的这篇文章吧。

网站精彩回眸

http://www.csdn.net/news/newstopic/ID前两位/ID.shtml

女软件工程师在各大名企求职80天的经历	ID：23111
一个程序员的酸楚感受：谁来关怀程序员？	ID：22986
脆弱的国产电子政务平台：达梦数据库在发布会现场接连发现BUG	ID：23624
Java之父：为Java发展惊奇　和脚本语言走得更近	ID：23364
经CSDN推荐荣获本次微软MVP的网友名单	ID：23432
40次面试被拒　上海交大学子为何成拒无霸	ID：23805
C/C++程序员需求量居高不下　传统语言魅力依旧	ID：23052
下一代DirectX——WGF几近完工　DirectX将不复存在	ID：23512
Eclipse3.1挑战百万下载　大受欢迎导致服务器崩溃	ID：22987
毕业&求职&就业 讲讲大学生七月里的困惑	ID：23235

软件项目经理必备素质

http://www.mypm.net/articles/show_article_content.asp?articleID=7972

我通过担任项目经理期间所遇到的种种现象，来总结项目经理所必需具备的素质，当这些素质您不具备的话，就需要花费多年的努力来培养它，如果无法培养成功，那么请您转换岗位，因为项目经理不适合您，您难以在这个方面获得成功。

软件项目质量管理实战总结

http://www.mypm.net/articles/show_article_content.asp?articleID=7894

本文详细阐述了作者对软件项目质量管理的认识，是作者实际经验的总结。主要内容包括对软件项目质量管理理论的认识，软件项目质量管理在实践中的具体做法。

软件开发中项目人员的心理分析

http://www.mypm.net/articles/show_article_content.asp?articleID=7896

“软件业，以人为本”。参与软件开发过程的各类人员的共同心理表现，对整个开发过程起着举足轻重的影响。作为一名项目管理者，非常有必要对其进行一一剖析，找出心结，然后对症下药，加大沟通力度，提高开发效率。

如何划分项目经理和技术经理的工作？

http://www.mypm.net/case/show_case_content.asp?caseID=1456

我们公司是一个中型软件公司，大概有300多人，其中200多人做软件开发，目前我们公司的大部分项目经理都是技术强人，因为是技术出身，他们更喜欢冷冰冰的电脑，因为电脑相对人更容易控制。公司一步步地发展，领导们发现技术出身的项目经理慢慢不能满足市场的要求……

http://www.codechina.net/resource/gotores.php?articleid=ID号

ID:140	文件分割器（汇编）	ID:15610	Delphi速算24点扑克游戏
ID:2453	邮件发送程序（VB）	ID:15662	JAVA语言入门(CHM格式)
ID:4234	PB创德图书销售系统	ID:15688	SSL加密传输JAVA实例
ID:4475	自制图片格式转换和保存入库工具源码	ID:15831	faisunSQL自导入数据库备份程序3.7（PHP）
ID:4566	C#高级编程源码	ID:16747	PB实用小例(常用PB源码，值得参考)
ID:4834	PB中如何获得机器ip地址	ID:16749	LCD液晶点阵提取器（单片机开发资源）
ID:5192	LyNews凌云新闻系统	ID:16769	Delphi7编程150例
ID:8705	Jsp简单留言本(Mysql)	ID:16762	（JAVA）条形码商品管理信息系统
ID:15239	（ASP）学校新闻发布系统	ID:16766	（Asp）鸭鸭城市V4.0
ID:15434	直接读写XML文件	ID:16779	用Delphi实现Word文件的预览
ID:15534	PHP访问Access数据库文件		

让您不会错过最HOT的信息

文档地址:http://www.csdn.net/develop/article/ID前两位/ID.shtm
论坛地址:http://community.csdn.net/Expert/topic/ID前四位/ID.xml

文　档

主题:Eclipse SDK 3.1来了
作者:Javamxj
ID:72499
摘要:Eclipse SDK 3.1官方版发布了,下载主页是:http://www.eclipse.org/downloads/index.php。不过,由于下载的人太多,基本访问不上去了,过些时间可能会好些。

主题:一步一步搞定手机串口短信收发
作者:doodu
ID:71673
摘要:网上不少关于串口短信收发的文章,但因为AT指令的灵活性,各个手机厂家都做了"继承"和"派生",因此需要自己研究针对自己手机的AT指令。

主题:EXCEL读取与写入数据的最佳方案(个人认为)
作者:WindowsBoy
ID:71659
摘要:在以往使用Excel文件时,我一般认为是使用COM组件进行EXCEL文档的操作,这样写程序既费时又费力(因为EXCEL组件没有帮助提示),写的程序还是大篇的重复代码。

主题:"身份证号"的数字分析
作者:lovelyxc
ID:71626
摘要:在现代社会中,一个人总是离不开数字。人在社会上总要有一个身份证号码,学生在学校里读书一定会有一个学号,而且这些号码不都是一些无意义的数字。我写的这个程序是用来分析这些数字,并把其中有意义的意思表达出来。

主题:使用Flash彻底美化你的程序
作者:vcleaner
ID:72354
摘要:曾经在VCKbase(VC知识库)网站看到过一篇关于Flash的文章,叫做《在VC中使用 Flash 美化你的程序》,文章中介绍的是使用Flash控件在对话框程序中播放Flash文件。

主题:终于体会到了,在.net开发不用C#那纯粹是自找麻烦啊
作者:shidongdong
ID:72116
摘要:我是个Delphi fans,自从试了一下Delphi8,第二天就删掉了,bug太多,没法用,只好不太情愿地使用着C#。Delphi2005出来了,就又装了一个试用版用了一下,感觉不错,有点欣喜若狂,就准备使用起来,但是随着程序规模的不断扩大,越来越多地发现了至少是令人不爽的地方。

论　坛

主题:在迈入.net的门槛时,你们也有我这样的忧虑吗?
作者:binkingaker
所属社区:.NET技术 ASP.NET
ID:4099637
摘要:"将近一个月的.net经历后,自我感觉要做好ASP.NET,确实还有很多东西需要我去学习。在看书过程中发现很多东西的理解是云里雾里,有时候甚至头大。有时候自己认为,我太笨了,根本不适合做程序员……"

主题:比我高学历的同事老是在找机会贬低我,怎么办?
作者:ok1996
所属社区:Web开发ASP
ID:4106761
摘要:学历和能力没有直接关系,这一点我们都很清楚。而倚仗学历高就看不起人的行为,我们自然会给予强烈的鄙视。

主题:北京,深圳,上海的程序员,月薪低于2000的程序员跳槽吧!
作者:ok1996
所属社区:Web开发ASP
ID:4151641
摘要:"让BOSS总以为IT界的劳工就值这个价钱了,会导致整个行业的薪水跟不上经济的发展,中国这几年不是报道经济增长都在8%以上吗?这几年我们的工资有这个涨幅吗?"

主题:失业两个月,赶问路在何方?
作者:fujingn188
所属社区:C/C++ C++语言
ID:4135204
摘要:找工作最重要的还是要有足够的实力做保证……

主题:从公司旧程序中发现了历史上最强悍的、佩服得五体投地的代码!
作者:cuixiping(无心)
所属社区:Web开发ASP
ID:4157350
摘要:写代码时一定要注意命名规范,尤其是企业项目开发,不要写出没人能看懂的天书来。

主题:提问:高级程序员与初级程序员的区别是什么?
作者:Kevia(不懂就问)
所属社区:.NET技术 C#
ID:4112823
摘要:初级程序员?高级程序员?区别应该不只是一张证书而已吧。

主题:南北经济实力对抗,哪里最合适程序员生存?
作者:leiaming
所属社区:.NET技术 C#
ID:4109238
摘要:你是程序员吗?你在南,还是在北?

为了拯救一个程序员的生命

文/贾菡 崔轶

浅绿，象征着生命的活力和希望；红色，代表着温暖和人们的爱心。从今年年初，由



这两种颜色作为基色并配以“拯救王俊”字样的Logo突然在很多人的MSN头像以及他们的Blog中频频出现。他们在用自己特殊的方式，传递着对一个程序员生命的关爱，用自己的方式，唤起更多人对他的关注，用爱心共筑一道坚固的城墙。

王俊，今年27岁，是北京北方银证公司项目经理，也是北京Java用户组BJUG（Beijing Java User Group）的核心会员，ID为后山。作为一个年轻人，他热爱自己的工作，有无限美好的前途，他也喜欢同志同道合的朋友们交流技术。在业余时间，他喜欢在自己的Blog上与大家分享他的经历和感受。此外他还有一个并不富裕但很温馨的家，一切看起来都那么美好。然而，正如莎士比亚所说，悲剧就是把一个非常美好的事物破坏给人看。王俊在今年春节期间接受了牙医的建议，全面检查了身体。但是，当他看到诊断单上赫然写着“骨髓增生异常综合症”（白血病的一种）时，一切都改变了。

不幸中的万幸，王俊在最适合治疗的时间找到了最合适的骨髓配型（他的妹妹），应该说通过骨髓移植，现在的医疗水平完全可以让这个小伙子康复起来。剩下的难题就集中在钱上。20万的手术保证金，加上30万到100万的手术医疗费，再一次把这个出身农民家庭的小伙子推到了死亡的边缘。觉得希望渺茫的王俊，决定默默地从朋友们的视线中淡出，准备自己去独自承受，面对这个残酷的现实。

然而朋友就是无时无刻不在互相关心的群体。半年来，王俊的Blog一直没有更新也没有参加BJUG的各项活动，这种反常行为引起了BJUG其他成员的注意。就在人们纷纷猜测各种缘由时，突然爆出王俊罹患重病的消息，大家都惊呆了。

不过，很快他们就决定一定要尽力帮助我们的程序员兄弟。BJUG的成员迅速展开了对王俊的拯救活动。他们先是创办了拯救活动的专门网站（http://befresh.bjug.org），并为活动设计了Logo；还在多家银行开立了捐款专用的账号；联系了多家网站添加拯救活动专门网站的友情链接，以扩大声势……

CSDN技术主编熊节也是BJUG的成员，看到这条消息后，感到要利用CSDN来做一些宣传。除了他自己将获得的两本书的稿费捐出来，他还准备说服所有合作的作者，一同将全部稿费捐献出来；同时他还将拯救活动的消息传播给在JavaWorld.tw网站的林上杰等一些朋友，获得他们捐献的书稿，并由此在海外展开了声势浩大的宣传工作。

2005年6月27日，CSDN网站正式向社会发布了“献出一份爱心，共同拯救重病程序员王俊”的倡议，并公布了接收救助汇款的公开账户。为了让拯救活动更加正规，也为了避免社会上不良因素的影响，CSDN以企业的身份担保此次拯救活动的真实性和消息的可靠性。

为了帮助王俊筹集医疗费用，《程序员》技术主编孟岩向所有本刊作者发出了倡议书，号召大家把稿费捐献出来，为拯救程序员兄弟的生命尽一份力。为了扩大宣传，引起更多人的关注，《程序员》和CSDN还借助更多媒体的力量，唤起更多人的关注，让更多的程序员来帮助程序员自己的兄弟。

CSDN的宣传声势波及到了国内最大的IT专业评论网站Donews，Donews也很快在网站上对王俊的拯救行动进行了宣传和报道。在这之后，很多网站都对王俊拯救行动的报道进行了转载，继而引起了新浪和《竞报》的高度关注，也加入了对该活动的报道行列中。

在BJUG的推动以及CSDN和其它媒体的大力协助下，很快许多Blog首页上的显著位置，都出现了拯救王俊的Banner，各方的汇款也纷至沓来，这让拯救活动的参与者们看到了希望。

现在，拯救活动募集来的钱已经达到了8万多元，加上来自王俊亲友6万元的借款，目前账户中总额达到了14万，再加上他自己原本借到的一些，已经基本凑足了20万元手术押金。现在，王俊也看到了拯救活动的顺利进行，原有的消极慢慢消除，笑容又重新浮现在他的脸上。最近，王俊已经向医生预约了手术，很快就要进入无菌仓，开始手术前的准备。

希望在拯救活动中继续延续。朋友是一种相遇；朋友是一种相契；朋友是一种相伴；朋友是一种相助。人生总有一些瞬间，让人温暖得泪流满面。为了拯救我们程序员的好兄弟，大家都在尽自己的努力。让我们每个人都在心中默默不停地为王俊祈祷，祈望上天能够对他多些垂怜和爱护，让他远离死神。让他少些病痛的折磨，多一些健康与快乐，让他从悲观失望中走出来，多一些信心与勇气，至少还有这么多程序员的兄弟在关心他，帮助他。

为了能够最大限度地扩大宣传的影响力，募集到更多的款项帮助王俊，2005年8月6日，由本刊和CSDN及BJUG共同承办的“Java十年技术趋势活动暨为王俊募捐活动”将会举行为王俊募捐的活动。希望能通过大会扩大拯救活动的影响力，并通过现场募捐的方式为王俊筹集更多的医疗费用，也力争通过此次活动让更多的开发人员或相关从业人员关注自身健康。



■ 策划 / 本刊编辑部

IBM软件十年之旅

以硬件著称的IBM公司，不但将触角伸入软件世界，而且已经对全球软件格局产生着越来越大的影响与带动作用，在IBM软件集团成立十周年之际，软件产业的IBM现象，已成为一个不得不说的话题。

知天命者以豹变图强

——IBM软件技术十年评述

文/黄甫

如果以开发人员的关注程度来衡量一家公司对于软件技术的影响力，IBM显然在过去十年取得了长足的进步。十年以前，在IBM软件集团成立之初，除了极少数如修道士般深居简出的大型机开发者之外，在中国似乎没有什么人需要关注IBM在软件开发技术方面的所作所为。IBM，这个几乎等同于计算机工业的名字，与上世纪九十年代初期PC软件的淘金热潮完全无关。然而十年之后，越来越多的中国开发者把目光聚焦在IBM身上，特别是IBM在开源方面的贡献更获得了开发者们质朴而热情的欢迎。这种热情直接反映在IBM developerWork网站上——如今这个网站已经成为最受中国开发者喜爱的技术站点之一。种种一切表明，IBM用十年时间重新成为了软件技术版图上的主要力量。

由于这一切发生在IT产业的领袖IBM身上，似乎没什么值得大惊小怪。与其说这是一个关于软件技术发展的话题，倒不如说这是一个关于几个大人物的商业故事。

是这样吗？

如果你了解我们这样的开发者，就会知道我们不可能让这样的观点不经挑战而传播开来。对于我们来说，这是一个关于技术的故事，这个故事的核心是，一家大公司在一次短暂的挫折之后，终于认识到了技术变革的大趋势，并且努力地改变自己以把握这一趋势，最终获得了成功。在这个故事中，技术变革的女神一直在舞台中央等待着舞伴。大象只是因为成功地握住了女神的手，才能够重新翩翩起舞。开发者的掌声是献给双方的，因为我们看见了女神，而其他人只看见了大象。

1.

事实上，在IBM软件集团成立之前，IBM就一直是世界上最大的软件供应商，同时也是软件技术的领跑者。这一点即使是在IBM处于低谷的1990年代初期仍然成立。例如在PC操作系统方面，IBM主导开发的OS/2被广泛认为技术上领先于同时代的Windows 3.0。在面向对象技术方面，IBM开发了VisualAge Smalltalk，获得了技术人士的广泛尊敬。同时，IBM积极参与了雄心勃勃的Taligent项目。围绕这些项目的实践使IBM的研发人员在面向对象技术方面积累的深厚经验，并且直接产生了影响深远的设计模式思想体系。在组件技术领域，IBM开发的SOM（System Object Model），即使以今天的观点来看仍然是最先进的组件技术体系之一。然而，尽管IBM在诸多具体的技术领域保持了领先，但是由于它没有能够把握住战略性技术变革的脉搏，这些令人印象深刻的技术成果却没能在商业上取得成功。

所谓"战略性技术变革"，就是二十世纪八十年代出现的以个人电脑为标志的计算模式的变革。在前PC的大型机时代，IBM是绝对的统治者，它以自己的大型机为核心，面向客户提供从硬件平台到系统软件，从应用软件到专业IT服务在内的完整方案。硬件是核心，软件只是附庸。这种带有浓重的大型机时代痕迹的、自给自足的大一统业务模式，在二十世纪六、七十年代天下无敌，但是在PC时代就显得非常不合时宜。PC出现之后不长时间，就已经被证明是将改变计算模型的新工具。在很短的时间里，大量以前没有机会接触计算机的企业和个人，将以PC为工具进行工作。总体上来看，企业中主要的计算任务将由PC来承担，主要业务数据将存储在PC的文件中。而随着PC软硬件平台的标准化，作为直接向用户提供服务的产品，软件将摆脱依附地位，成为独立的产业。

然而，即使在PC爆炸性发展的1980年代中期，IBM仍然没有能够意识到这些，在软件开发方面连连受挫。之所以出现这种情况，首先是因为IBM没有能够建设一条符合时代要求的软件生产线，从而无法将自己在软件技术上的优势转化为市场上领先的产品。一位资深的IBM软件技术专家回忆说，当时IBM的软件开发流程完全是基于大型机开发经验制定的，笨重而缓慢，充满了繁文缛节，无法适应PC时代软件开发快速、灵活、多变的节奏。这种马德里宫廷式的软件开发模式，不仅拖累了IBM自己的项目，而且拖累了其合作伙伴。1980年代末和1990年代初IBM所参与的一系列软件项目几乎都以失败收场。其次，IBM将大型机时代产生的

傲慢和封闭带到了PC软件世界，完全没有认识到开放已经成为时代的主流。当微软免费分发自己的Windows SDK，努力鼓励开发者为Windows开发应用软件的时候，IBM还在向OS/2的开发商收取费用，这种自我封闭的做法显得与时代格格不入。此外，面对生机勃勃的PC软件市场，IBM实际上一直都没有能够找准自己的定位。他们在最有利的时机错误地放弃了操作系统的控制权，此后十多年却又徒劳地企图挽回这个错误，同时又在很多不同层面上的技术和项目中分散着精力。人们始终不知道IBM究竟想在PC软件产业链中充当什么角色。这些失误使得IBM在技术上的卓越能力无从发挥，很多原本意义重大的技术创新不但没有给自己带来应有的价值，反而被竞争对手仿效而大获其利。



到1990年代初期，IBM终于被自己一连串的失误逼入困境。一个在十多年前看上去无比强大的技术巨人，竟然会陷入这样的困境，这当中固然有很多其他的原因，但是没有能把握技术变革的大趋势，显然是最重要的因素之一。

这一段历史对于IBM的影响显然是非常巨大的，也许最重要的一点，就是使IBM认识到，技术变革的潮流不是任何人所能阻挡，不论你多么强大，面对技术革命，都必须积极主动地对待，并且努力改变自己以适应潮流。这一点，没有人比1993年接任IBM CEO的郭士纳理解得更为深刻。虽然郭士纳并没有技术背景，但是他却能比大多数技术专家更清楚地看到，世界正处在下一个计算模型变革的门槛上，关于技术的一切都将面临改写的机会。

2.

1993年，虽然很少有人见过浏览器和Web，但Internet的主干已经成熟。在克林顿政府的高级官员中，“信息高速公路”的说法已经流传开。对于大多数技术人员来说，Internet只是一种信息的发布与共享通道，虽然很令人兴奋，但是没有多少人会认为这与他们的软件开发模式有什么关系。但就在这个时候，郭士纳和他的IBM高级管理层已经开始认识到，Internet将最终导致计算模型的又一次变迁，即以PC为中心的计算模型，将转变为以网络，确切地说是以Internet为中心的计算模型。PC时代行将结束，Internet时代即将到来。

IBM的最高管理者和技术专家们很清楚，这对于IBM来说意味着一个难得的历史机遇。首先，新的计算模型意味着Internet网络中的主干计算机必须承担繁重的计算和海量信息处理工作，这远远超过PC的能力，只能由大规模的计算系统来承担，而这正好是IBM发挥传统优势的地方。其次，在Internet为中心的计算模型中，终端设备可以多种多样，PC将只是作为诸多Internet设备之一而存在。PC时代的结束将导致整个产业重新洗牌，这对于IBM来说显然是一个机遇。第三，Internet时代的应用软件整体结构将发生根本的变化，新时代的大型应用必须依靠一个高性能的中间件软件来提供各种高级服务，这显然创造了一个巨大的新市场。第四，Internet将开启一个随需应变的电子商务时代，在这个时代里，企业需要高质量的专业IT服务，这显然

也是IBM的一个优势阵地。

这一切在当时也许远不像今天这样清晰，但是郭士纳和他的目光远大的同事们显然已经意识到了Internet技术变革的重大意义，并且提前开始了行动。正如郭士纳所说："计算模式的变革总是10-15年发生一次，当变革发生时，那些抓住了机遇并领导变革的企业将战果辉煌，而其他的公司将只能围绕这些主导型公司旋转。"IBM要在这一波变革中占据主导地位。

怎么办？

惟有以变图强！

一旦决定改变，IBM表现出来的巨大决心和强悍的执行力令人印象深刻，从1993至1995年，IBM实施了一系列重大的改革，雷厉风行，干净利落。IBM软件集团正是在这种背景下，于1995年1月成立的。特殊的历史背景使得IBM软件集团一开始就充满了使命感和面向变革的激情。软件集团的成立，标志着IBM将软件作为一种独立业务看待，彻底结束了软件的附庸地位，整合了其强大的软件研发力量。更重要的是，IBM从此可以按照软件生产的规律办事，摆脱大型机时代软件开发慢三拍的积习，快速提高软件的开发效率和质量，从而着手解决技术成果向产品转化不利这个严重的问题。经过长期的实践，IBM形成了自己独特的集成产品开发流程IPD（Integrated Product Development）。这种开发过程的最大特点是，负责产品化、市场推广和技术支持的人员从产品的构思论证阶段就参与进来，并在研发过程中全程跟踪，从根本上确保技术与市场的结合。

1995年7月，IBM软件集团并购Lotus公司，这一事件在软件开发社群看来具有格外重大的意义，首先这标志着IBM重返软件市场一线的决心，其次Lotus的Notes套件是前Internet时代最具Internet特征的天才级软件产品，IBM的行动表明了以实际的行动握住技术变革之手的决心。如今，大多数经历过这段历史的IBM老员工，都会自豪地讲述IBM重塑辉煌的因素：强有力的领导力，主动变革的决心和行动力，以客户为中心的彻底服务的理念，以及无以伦比的深厚技术积累。一旦技术创新通过高效生产的齿轮与市场相连，技术研发就再也不是一个负担，而迅速转变成为驱动公司获得成功的强劲动力。

可以说，IBM软件集团的亮相之舞是精彩的，它本身是审时度势的产物，同时又体现了变革时代的激变精神。天时、人和已备，现在，它需要一对腾飞的翅膀。

3.

1995年3月，Sun对外公布了一个名为Java的技术。虽然在随后的一年中，Java以史无前例的速度获得了工业界的接受，但在当时，大多数人还无法准确地描述Java在未来计算世界中的地位。也许除了IBM的一些决策者和技术专家之外，没有人意识到，Java真正的用武之地是在服务端企业级应用上。

从七十年代起，IBM在自己的多个商用大型机项目中，就广泛使用了虚拟机化的软件体系结构。例如著名的AS/400（后称eServer i系列），使用C/C++开发了一个虚拟机层，将硬件平台的细节完全封装，从而向上提供了一个完全面向对象的、拥有巨大线性地址空间的虚拟机。这种在当时十分具有创新性的软件体系结构被证明是十分有益的，它不但使高质量软件的开发和测试变得更加容易，而且隔绝了硬件层的变化，为移植和硬件升级带来了巨大的便利。

任何一个了解Java的人立刻就会看出，这一体系与Java非常相似。因此，很容易理解，当Java刚刚在市场上红火起来时，IBM比任何人都更清楚Java作为主流商用系统开发语言的潜力。另一方面，Internet架构上的分布式应用系统，规模和复杂度较之先前有大幅度的提高，必须采取基于组件的方法，软件开发的效率和质量才能得到保证。而在组件技术上IBM同样有多年的积累。

对于Java自己的发展来说，寻找正确的道路着实花费了一些时间。直到1998年J2EE问世，才终于为Java进军大型企业应用系统开发领域铺平了道路。然而，从旁观者看来，IBM在整个过程中都走在了Java的前面，以至于当J2EE将Java的主战场搬迁到企业应用领域时，IBM倒象是已经在这里恭候多时一般。时至今日，人们都很清楚，虽然Java是一个开放的平台，但是其背后强有力的支持者正是IBM。

IBM早就认识到，在Internet时代的计算模型中，在操作系统之上、应用系统之下，有一个规模巨大的中间件层，这个中间件层将承担起类似Internet操作系统的职责，对下屏蔽各个操作系统的技术差异，对上提供直接面向分布式应用的高级服务，因此在Internet时代成为基础的计算支撑和业务整合平台。于是，IBM在VisualAge的技术遗产基础之上开发了面向中间件市场的WebSphere。WebSphere是如此的成功，以至于今天当我们来界定中间件的概念时，很难不以WebSphere来作为参考。IBM在中间件技术和市场领域的成功，不但改变了当今的软件开发方式，而且也标志着IBM对于自己在新时代软件技术版图里的位置做出了清晰而明智的界定，把自己放在了技术链和产业链的一环上，既保证了自己的商业利益，同时又为与大多数厂商建立竞合关系奠定了基础。在这种基础上，IBM更进一步，积极投入开源运动当中，从而将整个软件产业的面貌彻底改变了。

事实上，IBM软件集团认识到，既然自己的定位是要做最强的中间件厂商，那么对于客户来说，底层操作系统的选择决策就主要取决于成本而不是技术因素。Linux成本为零，如果能够帮助Linux系统不断完善以适应商业应用的需求，

并提供完善的服务,那么Linux将变成能够为客户带来价值的产品。既然IBM本身不打算再涉足PC操作系统领域,那么支持Linux就成为最合理的选择。

不管IBM进入开源领域是出于什么动机,当它走出这一步的时候,就不仅改变了软件产业,也深刻地改变了它自己。IBM曾经是IT产业中封闭的典型,建立了一个壁垒森严的私有技术体系。一个最令人触目惊心的案例,是在1970年代因为PL/S语言而对RAND公司发起的诉讼。PL/S是IBM发明的专门供自己使用的编程语言,绝不外传。后来RAND公司根据IBM产品中公布的PL/S源代码,自己开发了PL/S编译器,这一行为本身是技术胜利,并不违法。但是IBM想尽办法,指控RAND在开发PL/S编译器过程中使用了未经授权的说明文档,从而逼迫RAND收回这一产品。由此可见,IBM曾经是一个多么封闭、多么以自我为中心的企业。因此,当1998年IBM决定将自己价值数千万的VisualAge for Java捐赠给开源组织的时候,人们确实很难不为这个企业在文化上发生的巨大变化而感叹。而几年之后,当初VisualAge for Java播下的火种,经过开源的热风,已经以Eclipse的名号成燎原之势的时候,IBM又先后向Apache组织捐赠了应用服务器Geronimo和数据库Derby,播下新的火种。这已经不仅仅是一种商业或者技术上的慷慨了,而是从根本上表明IBM软件集团已转变为全新的开放型技术企业。

Java和开源,就像IBM软件集团的两只翅膀,深刻地改变了IBM,也深刻地改变了软件产业。

4.

IBM因为没有把握住PC时代的到来而一度陷入困境,又因为成功地把握住Internet时代的脉搏而大放光芒。未来呢?IBM是否能够在下一个时代里继续领先呢?这就要我们冒险来展望即将到来的下一个计算时代。

总体而言,目前的世界仍处于向Internet时代全面过渡的时期,我们还要在这个时代待上几年。然而,随着Internet带宽的大幅度提高,Internet联入设备的日渐丰富,IPv6的逐渐应用,无线网络的普及,我们必将逐渐进入Internet时代的高级阶段。在这个阶段,计算模型又将发生一次质变。首先,计算设备品种的大爆炸将使得跨越各层次平台的统一软件环境成为必需品。其次,主要的计算和数据性工作将由网络节点中的骨干计算机来集中承担。第三,搜索将成为人们使用计算机和网络的基本方式。我们看到,IBM已经为下一代技术做好了布局。

在信息处理方面,IBM是数据库,特别是关系数据库的发明者。几十年商用系统的开发和应用,使IBM比其他任何人都更深刻地理解"数据"的意义。IBM敏锐地注意到,未来无处不在的宽带Internet将把整个世界变成一个无比巨大的数据源,然而其中绝大部分的数据是存储在普通的文件,而不是经过规范化的数据库中。数据库技术前进的主要方向,应当是把这些不在数据库中的数据也管理起来,提供类似SQL的查询界面,把巨大的数据源转成巨大的数据库。IBM已经在这个方向上投入了大量的研究力量,如果取得突破,无疑将彻底改变人们使用计算机和Internet的基本方式。

在计算模型方面,Internet已经使越来越多的计算任务从桌面PC重新回到服务器端,这已经对目前的服务器计算能力提出了越来越高的技术要求,迟早会使目前一般的企业和服务器租赁提供商不堪重负。在主干网带宽成数量级提高的情况下,高性能计算能力可以作为一种服务,由最具专业技术能力的公司在网上提供。IBM在超高速计算机研制方面具有传统优势,因此在网格计算领域当仁不让地先行一步。同时,终端的多样化也呼唤一个统一的、可适用于不同尺度设备的软件平台,Java在目前已经充当了这样的一个角色,而IBM也在这个方向上投入了足够的力量。

在软件开发方面,软件系统规模和复杂性不断增大,要求应用软件开发和集成商将软件的开发效率和质量再大大地提升一步。要跨出这一步需要一系列条件的共同配合,特别是软件开发过程和整体体系结构以及基本开发模式的改进。IBM收购这个领域的领导厂商Rational只是一个开始。尽管在技术界对于模型驱动开发还存在很多怀疑和争论,但是IBM显然充满了信心。软件系统越来越复杂,对于用户也提出了过高的要求。如何让软件在功能越来越强大的同时,还能够保持易用性?这正是自主计算技术的目的。IBM在这个领域已经位于最前沿。

而在自己优势的中间件领域,IBM也跟其它的主要力量一样,正在积极地推广SOA体系。从更高的角度来看,Services就是分布在互联网上的松耦合组件,它是十几年来一直不断发展的组件技术在高速互联网时代的必然形式。IBM在这个方向上不仅有中间件技术的固有优势,而且还具有开放所带来的技术活力。

指导这一切技术进步的基本思想,在IBM看来是十分清楚的,那就是"软件即服务"。当互联网发展到充分成熟之后,软件与网络都应当隐入后台,逐渐在普通用户面前消失,而以服务的形式表现出来。服务是IBM理念中的核心,软件的这种发展方向,对IBM既是一个挑战,同时又是一个更大的机遇。毕竟IBM是把服务精神写进基因的百年老店,历史已经无数次见证了它在技术上的长远眼光,而从低谷中重新崛起的经历更使IBM像一个年轻人一样充满了变革的活力,IBM显然将在软件的下一个时代继续强大。

大象起舞，蚂蚁离开

文/熊建国

在一堆机器或建筑或其他不同的工作场景中，到处都隐藏着软件。

“中间件尽在其中，你看到了吗？”这是IBM公司关于中间件的一则比较经典的广告。这则广告的寓意是，虽然软件不可触摸，但却极大地改变了我们的生活！

绝不只是玩玩

“可能只是玩玩”，这是多年前人们特别是普通开发者对IBM公司做软件的评价。

其实，IBM公司软件的历史并不比硬件历史短多少，只是IBM在一开始对于软件并没有一个具体的、清晰的策略。软件业的发展也成就了IBM的软件战略。比尔·盖茨曾经说过“十年后，真正硬件的成本将下滑到几乎免费的水平”。软件和服务才是IT业发展的方向，也许到那一天就变成买软件，送硬件了。

在十年前成立软件部的时候，IBM就明白了软件不能单纯依附硬件来生存，IBM的软件部必须符合市场对于软件的要求，而不是对硬件的要求。

IBM大规模的收购软件公司是从1995年以35亿美元得到Lotus开始的，这一次大收购改变了IBM软件人的心态，即“软件部”必须进入当前的软件领域，而不是依附在IBM的硬件产品上。此次收购也同时向市场宣誓“我要进入软件市场”。

接下来IBM又收购了著名的系统管理软件Tivoli，2003年又将Rational招入旗下，加上IBM自主开发的WebSphere和DB2，今天，五大中间件产品线建构了一个强大而坚实的中间件软件实体。在分析了软件市场的需求、认识到自身软件专业化的程度之后，IBM把自己的软件应用方案定位在中间件集成，把公司定义成一个Middleware公司。这便是IBM软件的第二次变革，要构建一个以中间件为核心的软件体系架构。

IBM以中间件理念为核心构筑的五大软件产品家族（DB2、Lotus、Tivoli、WebSphere、Rational）空前完整，而且包含了数据库、BI工具、基础设施平台等多个部分，已经是一种介于“纯中间件”产品和套装软件之间，更自由、更丰富的“大中间件”范畴。目前市场上其他厂商的软件产品往往都只涉及特定的某个领域，而IBM则是从整体和大局上对这些领域所需要的软件加以分化，并再次整合。

软件的世界里不再只有操作系统和应用软件，中间件成为一个独立的软件分支，逐步从后台凸现到前台。尽管软件的收入只占16-17%，但已经成为获利最高的部门，占1/3。至此，人们逐渐看清了蓝色巨人的软件战略。

从1995年成立至今，IBM软件服务部门已经走过了整整10年。10年的发展已经使得软件服务部门成了IBM业务的生力军。如果将该部门单独划出，它将是世界上仅仅次于微软的第二大软件公司。

2004年的年报显示，IBM公司软件收入已经占到总收入的15.67%。

软件即服务

在大型机的时代，IBM毋庸置疑地被称为高科技公司，但今天，IBM更愿意被人们称为一家能提供高品质IT服务的公司。

我们不妨回顾一下历史。

那个年代的IT环境是怎样的呢？20世纪80年代，IBM PC的发布极大地刺激了软件业的发展，当时，叱咤风云的四大PC软件厂商正以其单一产品引领市场。Lotus的表格软件Lotus 1-2-3、微软的操作系统MS-DOS、Ashton-Tate的数据库dBASE以及WordPerfect公司的字处理软件，它们在各自的领域呈垄断之势，并为各自公司创造了巨大的财富。1985年Lotus的销售额达到2.26亿美元，在美国IT公司中位列第60位；微软1.64亿美元，列第78位。

那时，几台机器用Novel网组织在一起，就已经能够做出惊人的应用了，视窗操作系统的出现已经让个人电脑市场开始蠢蠢欲动；特别是对网络的憧憬，未来一切看起来都是那么美好。20世纪80年代和90年代是软件业腾飞的时代。80年代的PC革命和90年代的互联网浪潮，计算机的两次革命带动了软件业的飞速发展。

宾夕法尼亚大学，研究市场和技术的教授理查·沙龙认为巨型公司一般很难适应技术变革，但是IBM却是个例外。IBM选择在这个时间切入软件，可谓恰逢其时。

进入了郭士纳时代，经历过一些曲

折的IBM开始了调整。郭士纳坚信，IBM应该成为一个服务型企业，他率先带领管理层代表团访问了很多大客户，比如P&G。他广泛采纳客户的意见，经常动辄邀请200个大客户参加一个会议，为的就是想要知道客户需要什么。从此IBM变成了一家以客户为核心的价值整合型企业。

搞技术的人总是固执的认为，卖一件具有高科技含量的商品才是挣钱的硬道理，但非技术出生的郭士纳让IBM人明白了，“不是谁发明了什么，就一定能够把它做好的”。在硬件方面，郭士纳先后卖出了IBM的网络部门、IBM硬盘事业部，甚至今天卖掉PC部门也是这一理念的延续，尽管这些技术都是IBM发明的。

不卖技术卖什么？卖服务，这就是IBM的答案。而一切核心的服务是离不开软件的，因为软件即服务。

一如早期的Lotus 1-2-3一样，这是一个非常好用的办公软件，解决了很多用户的问题，所以广受欢迎，但是随着用户需求的不断改变，他们开始更强调沟通和协作。在这样的市场环境下，催发了Portal这样的技术，从而改变了以往办公软件的职能，从简单解决个人问题，到与人沟通和协作的高效，这个跨越也是在IBM手中完成的。

弱化两头，强调中间

IBM软件部门，除了真正意义上的应用程序以外，其他所有层面的软件，它都有所涉猎，这其中不但包含有数据库，处理业务逻辑（宽泛概念）的中间件，甚至于操作系统（这里不是指桌面操作系统，而是如AS/400或者Z/OS这样的小型机、大型机的操作系统）都是其产品。

IBM的软件也许不是最了解开发者的，但绝对是最了解客户的。当客户说需要具有协作能力的Office，IBM就收购了Lotus；当客户说需要加强网络管理，IBM就买下了Tivoli；客户说需要中间件，IBM又包装了所有此类的产品为WebSphere；当客户说需要管理开发流程以便更方便地分析企业业务时，IBM便收购了Rational。

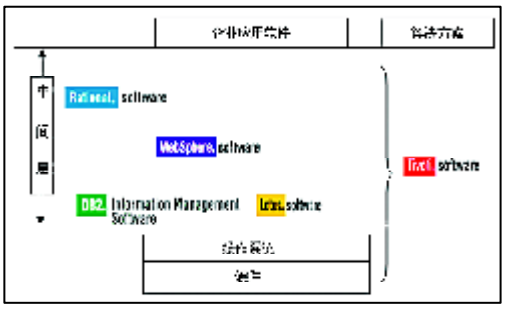
IBM的软件经过了10年的发展，在今天已经形成一个庞大的体系。许多人看IBM的软件都有一种雾里看花的感觉。经过近十年的收购，IBM一共买下了41家软件公司，所有产品形成了今天庞大的中间件王国。

而IBM搭建的软件王国和微软的软件王国又有着本质的不同。IBM的软件产品以更开放合作的姿态出现，对客户来说，当选择了IBM的产品，意味着你还有更多的选择。

IBM软件的核心战略：要做最强大的中间件供应商，这就是“弱化两头，强调中间”的中间件策略。IBM所理解的中间件是一种抽象的、大的中间件，是介于操作系统之上和行业应用软件之下的大中间层。

我们可以将这5大家族的产品分为两条线：开发线和管理线。

WebSphere家族是企业用于整合的中间件平台，与Rational软件家族同属企业开发线上的产品，Tivoli系列属于企业IT基础架构管理线，而Lotus产品系列可以说是一个企业内部协作的工作流平台，DB2则是IBM所有应用的数据支撑平台。实际上，经过半个世纪的发展，从底层到高层，包括硬件和操作系统等产品，IBM公司全有了。IBM现在的软件策略是弱化两端，即底层的操作系统和高层的企业应用软件，而彰显和强调一个庞大的中间层（IBM所理解的中间件）。



这五大产品线相互依存，能够随意组合IBM强大的解决方案。DB2系列来存放和分析企业的所有数据，用Lotus来完成人员的沟通和交流，WebSphere来完成企业流程的整合，Rational完成企业应用整个生命周期的开发和部署，而Tivoli系列则管理企业所有的IT软硬件基础设施，给企业应用提供安全的保障。

在软件领域，IBM还会做什么可能无法猜测，但IBM不会做什么却非常清晰。第一，IBM不会去做企业应用软件，或行业应用软件。在这一层，IBM提供的是咨询及服务，以及解决方案。IBM会和其他做行业应用软件的厂商一起站在客户面前，所以IBM和SAP等一些公司合作紧密，甚至专门为这样的公司来优化自己的产品，使这些产品能够更好地运行在IBM提供的中间件平台上。

第二，在底层，IBM不会去再做一个操作系统，因为IBM只需要自己的产品支持所有的操作系统就够了。当年，当IBM对Linux的支持如日中天成为Linux社团一道亮丽风景的时候，所有Linux的崇拜者们都仰视IBM，等待IBM抛出一个主流Linux产品。但IBM却并未如人所愿推出一个IBM自己的Linux，而只是让它的大部分产品都可以平滑过渡到Linux上。这一策略的确可以看出IBM的老谋深算，操作系统已经够多了，我只需要创造出一个客户可以自由选择操作系统的环境。这时，操作系统之上的中间层舍我其谁？

IBM软件的另一个特点就是完整性，这一特点在Tivoli产品线上表现得尤为明显。其实在收购Tivoli之前，IBM也有一些系统管理软件，收购后将这些产品合并在了一起。从当初单纯地做系统管理，不断扩展，到1998年的时候，扩展了存储管理的功能，2000年扩展到安全

管理。所以今天的Tivoli产品家族包括三个部分:系统管理(自动化)、安全管理、存储管理。因为IBM很早就认识到,单纯做存储管理或安全管理是无法满足企业需要的,而最近赛门铁克收购Veritas也是对IBM发展方向的一个佐证。

Tivoli绝对算是体现IBM公司IT服务和“随需应变”服务思想的主打产品,IBM的想法是,希望将你企业的所有IT基础架构(包括所有软件和硬件资源)统统管理起来,并且能够对资源做到随需应变的调度和使用,从而减少企业在IT上的投资。

IBM的软件总是希望全方位、多角度地解决企业的需求,非常强调产品的完整性。

比如,开发线当中的Rational系列,几乎可以算是IBM收购的软件产品中,改动最大的一个。当IBM收购Rational一年后,便推出了Rational Software Development Platform,去年到今年推出的一系列产品时,所有Rational的老用户以及开发者都懵了,人们纷纷惊呼:IBM到底对这个产品干了什么?代号为Atlantic的产品是IBM用了两年时间打造的Rational家族新品,当今年IBM拿着这一系列新产品全球巡讲时,人们突然明白了,IBM已经从非常理解企业客户演变为开始慢慢理解开发者了。

“像造车一样来开发软件”一直是开发者的梦想,Atlantic的推出和福特当初引入第一条流水线来批量生产汽车有些类似。当然他们做的事完全不同,福特是把它标准化,而Atlantic是通过模组化提高品质,并把开发时间缩得更短。这一切都由Atlantic的三大特点来保障:一是基于业界开放标准,包括Eclipse 3.0和UML2.0;其二是产品的易用性;其三是端到端。

这些变化不但符合IBM公司软件大而全的特点,同时也暗合了软件开发流程变化的趋势。今天的开发者已经由关心单个开发环节转变为希望掌控整个软件开发生命周期。其它厂商如Microsoft、Borland都开始将自己的产品线向软件工程的其它环节延伸,并开始朝着Rational这种大集成的方向发展。产品线是大集成的,但单个功能又可以是分开插拔进开发工具当中去的。

以数据库软件为例,当年IBM公司的一时疏忽造就了今天的Oracle。在数据库软件方面,IBM奋起直追,据Gartner今年5月份发布的市场报告显示,在关系型数据库市场,IBM占有34.1%的市场份额,Oracle占33.7%,两大厂商几乎战成平手。IBM的确不愧在关系数据库方面有深厚的技术底蕴。

因为IBM常常和客户一起来开发他们需要的软件。在DB2的8版本研发前夕,IBM调查走访了大量的数据库客户,对他们最关心和希望的功能需求进行排队分析,发现70%的客户对数据库的健康诊断非常关心,对繁琐的日常维护管理工作和数据库性能调节极其厌恶。因此8版本的研发重点就放在了“自主计算,自主管理”上。

在产品概念上,Oracle推销的是“网格计算”,而IBM抛售的却是“自主计算,自主管理”。且不说网格的时代是否已经来临,但DBA管理数据库的繁琐而复杂的工作却是实实在在的。所以在概念的实用性上,IBM公司已经胜出一筹。数据库系统软件的发展必将更加智能化、自动化。

大象起舞,蚂蚁离开

实际上,在Eclipse和WebSphere Studio以及Rational系列产品之前,IBM的软件在开发者心目中的形象是难以言说的。5年以前,要一个开发者钦佩IBM是困难的,他们会将更多的崇敬献给Borland,甚至是微软和Sun这样的公司。

从开发工具的层面上讲,IBM的确没有微软和Borland那样更懂得开发者,每一个小功能小控件都能挠在开发者的痒处。当IBM积蓄力量研发出WebSphere,当IBM对Rational进行大刀阔斧地改造,特别是当IBM捐献出开源项目Eclipse,开发者眼中立刻出现了一个“清新”、“友好而开放”的IBM形象,再也不是那个“僵硬笨重”、“不苟言笑”的蓝色巨人。

Eclipse在业界的影响超过了以前任何一款开发工具,其开放的可扩展的IDE架构甚至影响了许多软件开发的模式。于是基于Eclipse架构的IBM的开发工具WSAD一投入市场,很快便赢得了开发者的青睐,一扫当年VisualAge留给大家的不快。要知道许多人都有使用VisualAge的惨痛体验,笨重、庞大而复杂就如IBM自身一样。

而今天的开发工具WSAD的使用却完全是另外一副景象。它自带数百种商业或非商业的插件,程序员可以非常方便的来进行J2EE应用的开发。通过WebSphere的管理程序,可以很方便的进行应用部署等工作,从JMS到Web Service,WebSphere都有自身的一个系列的产品在运营和支撑着,从大到小,都为开发者考虑到,可以让你把主要的精力放在应用的开发上,完善的文档支持同时又能够让你充分了解它的特性。

为什么会这样?从这个产品看,IBM走到今天才算真正了解了开发者。IBM一定是顿悟到了什么东西,那就是软件开发的开放化、标准化。特别是在捐出价值4000万美金的Eclipse后,IBM几乎彻底征服了全世界的开发者。

今天的蓝色巨人,已经走进了开发者的生活当中,同时开发者也在影响着IBM的软件。越来越多的开发者意识到,除了了解微软的开发技术以外,现在必须要开始了解IBM的软件,而做到这一切,IBM整整花了10年的时间。

正如郭士纳在自己书里写道的那样,“问题不在于大象能不能跳舞,而在于一旦大象开始跳舞,蚂蚁就必须离开舞台!”。

而此时,在软件舞台上,大象已经开始翩翩起舞……

第三方观点

我眼中的IBM

善待合作伙伴是长久之计　　中科软总裁　左春

我个人认为,软件在IBM公司内部的地位要远高于其现在的营业额比重。这有两个因素:一是软件潜力高,利润或绩效好。另一方面,IBM将全球PC业务卖掉后,自然也更加重视软件,希望大幅提高软件的销售额。从合作伙伴的角度来看,这体现在原来IBM可能不太重视的小单子最近开始加大关注力度,同时,IBM也在推广一些半应用级的产品,如CM、Portal等。

IBM有着很全的产品线,甚至在各个产品线上都与很多公司存在竞争关系。与其他竞争对手相比,IBM最大的差别其实不在于产品的技术本身,而在于其销售策略、服务策略以及做客户、做事的方法。

IBM强调贴近用户,重视合作伙伴,要求"随需应变"。而"随需应变"的核心是按照市场和客户的需求发展,贴近价值链中的用户端,同时与合作伙伴共同发展。

庞大的产品线下,IBM有着完整的解决方案,但它的软件策略既有独立完整的方案,也有供客户选择的开放性的合作方案。这两年,IBM比较重视合作伙伴的价值,有时虽然也会冲到客户一线做销售,但它在这方面控制得比较好,让合作伙伴有利益,这是比较长久的一种做法。

作为一家大型的ISV,中科软与很多大厂商都有合作。从时间上来说,我们和IBM软件的合作开始较早,记得第一个合作的产品是MQ,那时市场上中间件的概念刚推出,很少有人了解,为此IBM专门派来国外技术专家作培训。目前,我们在各条产品线都有合作,除了销售合作外,IBM还经常与ISV进行技术交流。

整体感觉,IBM软件的总体策略非常正确,但在执行中由于内部分工上难免会有一些制约和消耗,在一些较大的项目上,感觉其内部的配合还存在一些问题。

总之,IBM软件实力相当强,ISV应该从双方合作中壮大自己,国内的软件企业也应该好好向它学习。

服务第一,软硬随后　　普元信息技术有限公司副总裁　程朝晖

IBM见证了整个IT业的发展,包括世界软件业的发展。但是在前任CEO郭士纳上台(1993年)前,对于IBM来讲,软件一直从属于硬件,是为了满足其硬件的发展而发展,因此历史上甚至于可以"怪罪"IBM,差一点将软件业扼杀于摇篮中。

1969年6月23日,美国司法部通过了著名的反托拉斯法案后,IBM不得不宣布不再免费随机提供软件,从而开始为其硬件和软件分别定价。那一天可以称为软件业的官方元年。80年代初,IBM委托微软开发其标准架构PC机的操作系统,由此促成了最大软件公司的诞生和PC的兴旺。虽然郭士纳上台前软件一直是非重点部门,但他的软件营业额却比微软、Oracle和CA的总和还多。到了郭士纳的服务战略开始起作用的时候,软件才渐渐地爬到了战略第二的位置,软件开始改变IBM的思维与文化。

1994年7月,IBM在上海成立了大中华区(Great China)软件总部,吹响了进军中国大陆及港台的号角,开始把全球先进的软件(OS/2、Lan Server、DB2、CICS、MQSeries、Lotus、WebSphere、Tivoli等)带入中国。这些占满了企业级计算环境的货架,同样也实现着IBM的"e-Business On Demand"梦想。

IBM软件的核心竞争力在于其企业级的能力和完整的电子商务软件平台。IBM全球有如此多的企业级客户,软件平台需求充裕,软件平台的验证自然也非常充分。

目前,IBM软件几乎覆盖了软件的全生命周期,需求、设计、编码、测试、运行和管理。五大产品线组成了完整的电子商务平台,构成了IBM的软件核心竞争力,再加上领头的服务策略和硬件的商务平衡,其他单个平台的软件公司要想与之抗衡,并非易事。

但从辩证观点来看,太多的历史、太长的产品线同样也会成为包袱。IBM要想发展一个新的技术和产品,往往因为要考虑众多已有客户的利益,而不得不放弃许多好东西,这就是平衡的代价。总之,五大产品线的组成无可厚非,只是未来IBM如何保持和纳入不断更新的技术和产品需求,这才是关键所在。

我认为,IBM与微软、BEA、SUN等企业间处于竞争与合作共存的关系,它的最大竞争对手是微软,这是Java与.NET两大阵营的竞争。假设Java失败,那

IBM的软件部就没了，随之IBM的硬件部也没了，可能最后剩下的就是服务部在为微软做咨询顾问了。当然这种情况近5年是不会发生的，因为客户需要Java。

IBM目前有三大赢利点：硬件、软件以及服务。但企业级随需应变的电子商务运行平台是IBM的定位，关键是运行平台。目前看，还是服务第一，软件和硬件随后，在现任CEO的手上看来是不会改变了。但不管如何，IBM的软件在全球的软件格局中还将继续扮演着重要的角色。

象之舞：IBM软件产品体系概览

文/吕翊

事情还得从一起“蛇吞象”的故事说起。2004年，中国的企业联想成功收购了IBM的PC部门及其部分品牌，实现了在国际PC市场的诺曼底登陆，业界一时哗然。这里我们不想对这笔收购案本身做出什么评述，还是让我们究其现象，探其本质。IBM之所以干脆利落的出售了其赖以成名的PC业务，完全是因为其整体利润的主要重心，已经转向了软件和服务。

要想了解IBM的整个软件产品体系，还是从并购这个词汇开始。当前业界不乏并购个案，诸如甲骨文收购仁科等，但是以个人的观点来看，这些案例都无法与IBM的若干并购案相媲美。通过一系列有规划、有预谋的并购，IBM获得了Lotus、Tivoli、Rational和Informix等公司的产品和技术，打造了一个空前完整的产品家族。这一远见卓识，给予再多的溢美之辞也不过分。

IBM的软件家族由五大产品序列构成，包括WebSphere、DB2、Lotus、Tivoli和Rational。这些产品线看似相对独立，事实上却体现着一个共同的用意——整合。

● WebSphere：流程的整合。通过对WebSphere产品体系的打造，IBM利用它作为其服务器端SOA架构整合的重要载体和工具。利用WebSphere Application Server打造了一个强大的J2EE中间件平台，而通过对MQ Series的有效开发，又实现了对业务应用端到端的连接和整合。应该说，通过WebSphere产品序列打造的“企业服务总线(ESB)”，无疑将成为企业级业务流程整合的利器。

● DB2：信息整合。应该说DB2数据库的应用在早先并不十分普及，但是随着IBM对DB2系统的不断投入和版本的不



断更新，IBM已经在数据库领域获得了长足的进步，DB2也由原先的数据管理提升到了包括UDB、II、BI和CM在内的全面型产品的信息管理。值得一提的是IBM对Informix的收购，明眼人应当能够看出IBM在数据库业务上的勃勃野心。

● Lotus：协同环境的整合。IBM公司1996年对Lotus的收购的确可以称得上是一个“双赢”的举措，不但救活了Lotus，也极大增强了IBM在群件和协同产品方面的实力。目前，Lotus系统仅邮件用户就已经在全球获得了超过1.1亿的用户。而通过其辅助产品Domino Workflow、Domino.Doc、Sametime、Quickplace、LMS等产品又实现了对工作流处理、文档管理、实时交互、团队协作以及在线教学的全面支持，应该说构筑了一个强大的一体化的协同环境。Lotus近期推出的Workplace产品序列，其面向客户端整合的WCT更是成为了IBM客户端整合战略的重要棋子。

● Tivoli：系统管理环境的整合。IBM的软件应该说更为关注企业级的应用，而就一个企业而言，无疑需要对其整个IT环境的系统级管理进行投资。IBM适时地看到了这一点，通过对Tivoli的收购和大力发展，无疑将为其在企业IT系统管理环境的构建方面获得更多的市场份额。借助Tivoli，企业可以实现对整个PMI安全体系的构建，可以实现对包括网络、主机、应用系统等IT系统的方方面面的有效管理，使得企业的IT部门能够获得一个集成化的整系统管理环境。此外，利用Tivoli和其他IBM产品的亲缘关系，又能够很好的实现对包括DB2、WebSphere、Lotus等在内的其他应用系统的更为有效的监控和管理。

● Rational：应用开发环境的整合。拥有了WebSphere、DB2、Lotus和Tivoli之后，IBM的软件体系似乎已经天衣无缝了。但是IBM还是一如既往地敏锐，通过对Rational的收购，IBM又拥有了与应用开发者相关的整合平台：Rational Rose。进行应用软件开发的程序员应该说没有人不知道Rose大名的，但是这朵玫瑰花最终又插到了大象的头上，让人们还是有些没有思想准备。通过对Rational Rose系统的有效融合和再造，IBM已经为应用开发者打造了一套集成化的、完整的、涉及整个开发生命周期的管理平台。特别是对于利用IBM其他业务软件进行应用开发的人们，更将享受到来自IBM Rational的“小灶”。

五大产品线，五方面的整合。IBM已经打造了一个可以触探到任何企业应用的软件王国，而且这些应用软件平台空前完整。它包含了数据库、BI工具、协作工具、开发工具、管理工具、IT基础设施等多种成分，已经形成了一种介于纯中间件产品和普通套装软件之间的，更丰富、更自由、更博大的一个“大中间件”产品范畴，开创了业界之先河。作为IT从业者，一方面我们看到了大象的曼妙舞姿；另一方面，我们也似乎听到了大象在软件业踏平一切的脚步声。

第三方观点

我眼中的IBM软件

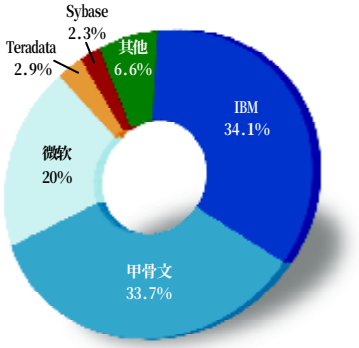
DB2 Information Management Software 遭遇激烈竞争 文/沈钧

历经20年的风雨，DB2已从最初单纯的RDBMS，发展为集数据库服务器、商业智能、内容管理、数据库管理工具以及信息集成为一体的卓越品牌。在全球500家大型企业中，几乎85%在用DB2数据库服务器。

来自外部的威胁

据Gartner Dataquest的统计，全球数据库市场在2004年增长了10.3%，总体市场增长主要来自：商业智能、数据仓库和数据分析。市场份额为，IBM 34.1%，甲骨文33.7%，微软20%，接下来是NCR Teradata占2.9%，Sybase占2.3%，剩下的6.6%由其他公司瓜分。

2004 年全球数据库市场分额（Gartner 统计报告）



总体上看，数据库巨头三足鼎立局面业已形成，IBM最大竞争对手为甲骨文（Oracle）。

整合理念亦非IBM所独有，另两大数据库厂商甲骨文和微软，也逐渐将商业智能、内容管理、数据的跨平台集成功能内置于其产品中。微软在1998年发布的OLAP服务，即是第一个集成到SQL Server中的BI技术。甲骨文在新推出的10g中，把只能跨硬件的群集技术提高到跨操作系统的网格运算群集。Teradata是数据仓库和商业智能领域的专业厂商，市场份额不高，但总体增长排名第一，得力于其专业性，对行业客户有较高的吸引力。

就技术指标上的比较，产品同质化已经形成（参考《Oracle数据库10g与IBM DB2 UDB 8.1技术对比》http://www.oracle.com/global/cn/documentation/10g/competitors/10gvsDB2.pdf，虽然竞争对手的报告有失偏颇之处），在网络计算、高可用性、数据仓库、商业智能上，两者不相伯仲，但IBM产品的应用开发GUI界面的易用性和人性化向来就不被称道，DB2亦未能幸免。对市场份额而言，目前Oracle与IBM DB2之间有0.4%的差距，反映到收入还不到3千万美元，还不能说明谁是赢家。

除了在高端数据库市场与甲骨文抗衡外，IBM还与微软面临竞争，而微软的SQL Server在单一平台（Windows）上的行情也被一路看好。其具备两大特点：一，足够智能化。在自动优化、自动管理方面非常到位，对开发者和用户而言，运行数据库时无需进行更多的优化工作。二，明显的价格优势。

如何应对？

面对Oracle咄咄逼人的进攻态势，IBM为了挽回颓势，赢回客户，在技术和战略上都作了准备。一方面，IBM寄希望于将在今年八月推出的下一代数据库产品——Viper的全球试用版。据悉，该产品不仅将包含Oracle现有的旗舰产品的技术——大幅提升数据的处理性能，还将帮助企业更好地实现流水线式的管理架构，并使开发者的二次开发变得更加容易。IBM认为，新版本中的XML功能，以及“多维群集”和“范围分区”的结合使用，都将使该数据库产品具有明显优势。另一方面，IBM对“随需应变”和“整合之道”这两张牌保持着持久的信心，结果如何？我们只有拭目以待！

IBM随需应变电子商务的基石—DB2 文/牛新庄

IBM的下一个收购对象是什么？没有人知道，但是人们眼中的这家传统的硬件供应商经过十年的发展和并购已经逐渐打造成了一个软件帝国，这却是有目共睹。IBM以中间件理念为核心构筑的五大软件产品家族中，DB2数据库是实现随需应变电子商务的基石，其强大的功能足以满足大型企业的需求；而其灵活的架构，又能适合中小企业的要求。

竞争惨烈的市场份额

关系数据库的原理是一样的，关系数据库经过二十多年的发展已经非常成熟，目前市场上主流的关系数据库有Oracle，DB2，Sybase和SQL Server。DB2数据库从1996年开始进入中国，而Sybase从1991年就已经进入中国。所以在2000年之前，DB2在中国市场的占有

份额非常少(不到7%)。但是随着IBM对DB2的推广发展再加上IBM在2001年对Informix的成功收购,如今DB2在国内的市场份额在快速直线上升,而国内一部分早期的Sybase用户很多都转向了DB2。目前国内的现状是,很多行业用户(金融、烟草等)都在使用DB2,而很多大中型企业在构建ERP时多选择Oracle, SQL Server则在中小企业有着广泛的应用,预计将来的关系数据库将是DB2, Oracle和SQL Server的天下。

产品技术特点比较

开放性/可移植性。DB2在各层面,从网络协议的支持到应用开发程序的编程接口,数据库存储程序及触发器,服务器之间的分布式运作,都是基于最新的国际标准,以确保其开放性及移植性。Oracle的许多功能都是使用其私有的、非开放性的技术。例如:PL/SQL语言是Oracle私有的。其存储程序语句也不符合ANSI SQL及ISO SQL的标准,客户机/服务器的编程接口,SQL*NET也不是根据标准定义的。Sybase的编程接口不是开放的,使用的是自己的OpenClient/OpenServer接口,而不是基于某一标准的接口。Sybase在构建存储过程时没有使用SQL92等相关的标准,因此编制存储程序和触发器所用的Transact-SQL也是其专用的。Informix私有4GL编写的程序只能局限在Informix上运行。SQL Server的应用开发仅仅局限在Windows平台上,它注重功能的日臻完善和易开发性,以及和微软自己开发平台的整合与集成。

商业智能。DB2 OLAP Server远比Express更易使用和实现。DB2 OLAP Server提供读/写能力,这对一些关键性的多维分析扩展应用(如预测和预算),是非常有用的,而Express只支持读功能。IBM的商业智能解决方案中,拥有数据挖掘(Intelligent Miner)工具。Oracle Express只支持读的能力,不利于支持预测和预算。Oracle缺乏数据挖掘方面的工具。Sybase数据集市(datamart)没有提供仓库(Warehouse)管理与自动化的功能。SQL Server在Windows平台可以实现商业智能,而且非常灵活方便中小企业部署数据仓库。

并行能力。DB2采用DB2 ESE(DB2 EEE)实现并行操作,Oracle使用RAC(OPS)技术实现并行处理,Informix使用XPS实现并行。Sybase和SQL Server没有成熟的并行模型。在并行能力方面Oracle通过共享存储实现,而DB2通过NFS实现,相对来说Oracle的并行模型更加成熟。

并发控制(锁)。DB2很早就实现了行级锁,DB2的锁面向的角度是"读一致"性,但是有锁"升级"和锁等待现象,会影响应用并发。Oracle利用意向锁及数据行上加锁标志位等设计技巧和回滚段来保证能读到数据的前"影像"(before image),Oracle没有锁升级现象,Oracle的锁面向的角度是"可读性",但是有可能会造成"脏读"。Sybase,Informix和SQL Server数据库在早期还有"页"级锁,直到近年才实现"行"级锁,也存在"锁升级"现象,它们的并发能力要比Oracle和DB2弱。

未来的发展方向?

随着IBM随需应变电子商务的发展,DB2将着力在"自适应"、"自诊断"和"自调整"方面满足客户自主计算,自动化管理以适应随需应变。而最新的Oracle 10g在网格计算,RAC和Data Guard方面做的非常完美。将来的DB2数据库将在高可用性、安全性、电子商务、数据挖掘、XML和应用产品支持方面大力的发展。

结言

总的来说,DB2数据库在并行处理能力、在开放性/扩展性、海量数据处理、电子商务、JAVA支持和商业智能等方面在这几种数据库中是最好的。DB2的优化器是公认为关系数据库中最好的优化器,但是DB2在应用开发、图形化界面、用户友好性和简便性、并发方面需要加以提高改进。每个数据库都有自己的特点和优缺点,很难说哪一个数据库是最好的,企业只能根据产品的特点和优势结合自己企业的实际情况选择最合理的数据库。

WebSphere software 且行且唱 文/马伟

10年前,IBM软件集团成立,3年后他们发布了Web战略的核心产品WebSphere。迄今为止7年过去了,WebSphere走过了不寻常且艰辛的道路。目前已经成为市场占有率最大的应用服务器,市场占有率从最初的11.7%提高到2003年的41.3%。

EBOD的基石

WebSphere并不是一个单一的应用服务器,而是一个基础架构软件平台,具有一套集成化的开发工具、门户、商务和集成软件。成为一个Web应用架构的必要组成部分。在它的产品家族中,涵盖了完成中间件功能的全部产品,所有的产品皆以WebSphere Application Server和MQ Series为核心,包括开发工具、表示工具、部署工具,以及众多的快速应用开发工具包。通过使用WebSphere家族产品,可以快速地开发出符合要求的应用产品,并且保证应用产品的安全性、伸缩性、易维护性。

WebSphere是IBM随需应变电子商务战略的五大重要成员之一,作为EBOD的核心和底层基础,WebSphere与协助安全及组态管理的Tivoli、协同的Lotus以及DB2信息管理的多种技术集成在一起,以保持彼此功能上的一致。而Rational则强化了IBM在Java、C及COBOL等上面的应用开发能力。WebSphere就像一块稳固的基石,支撑起了IBM这栋电子商务解决方案大楼。

在路上

WebSphere已经成为ISV的应用平台首选。超过9000家ISV在WebSphere上编写应用软件。IBM是Eclipse的开放源码技术的积极倡导者和参与者，并且编写、开放了其核心Work Bench的源码。业界对Eclipse的追捧是有目共睹的：它的下载次数超过200万次，来自多达125个国家；175家工具开发商基于Eclipse开发工具和部件产品。在应用服务器市场上占有超过40%的份额，具有10%的市场增长率，目前在金融领域和电力行业的市场占据雷打不动的地位。

WebSphere与WebLogic

WebSphere目前已经广泛应用在为跨各种计算平台而建立、运行和集成的电子商务应用中。它具有基于J2EE、XML等开放标准和新的Web服务标准，并具备IBM在可靠性、可扩展性和安全性上的主要优势。同类产品中最具有竞争力的就是BEA WebLogic了。

WebLogic没有WebSphere中的管理进程和“保姆”进程。从体系结构上说，WebLogic Server与WebSphere Application Server相比有一定的差距和不足。一旦用户的服务器出现非正常的错误引起应用服务器的退出，那么用户再访问应用服务器的时候将出现错误。这点WebLogic Server对于容错的处理能力远逊于WebSphere Application Server。

BEA作为“从收购中获得增长”的公司，5年中他们收购了超过25个项目，如此众多厂商的产品被整合在一起，一方面，用户不得不面对不同的厂商，另一方面，用户也不得不对如此众多的产品的集成性产生怀疑。而IBM以五大产品线的庞大家族出现，为客户提供整体解决方案。

WebSphere的未来

WebSphere将会致力于Web Service的发展，WebSphere未来将会推出On Demand Service Grid，这是从Services角度出发的网格计算技术。简单地说，就是用Web Service作标准，让不同的网格可以去访问各种各样的Web Services。此外WebSphere提供的Web Service还可以直接调用DB2的Store procedure。

WebSphere会在IBM EBOD战略中发挥更加重要的作用，一路且行且唱。

软件开发的理性天堂 Rational software

文/徐锋

IBM收购Rational！这条新闻曾经深深地震撼了业界，从最大的用户成为拥有者，这无疑是对Rational产品体系、发展趋势最大的认可。

完整的产品体系

Rational具有完整的产品体系：以完全面向对象的思想组织，和工业级标准建模语言UML融会贯通，利用在Objectory基础上发展演变而来的Rational Unified Process作为基础，形成了集需求管理、分析设计、项目管理、配置管理、测试与质量保证、文档生成与一体的完整的软件工程软件体系。



一直以来，Rational都是软件工程管理新思路、新实践、新方法的发源地，成为了这个领域发展的引导者，这应该也是其最重要的竞争优势之一。

大象与狼在博弈？

IBM被郭士纳喻为“大象”，这与Rational的特点不谋而合，也许这也是IBM与Rational能够喜结良缘的主要原因之一。万物均有两面性，体系完整性给产品带了巨大的优势，同时也必然使得个性化、专注某一局部方面存在一定不足。

● RequisitePro VS DOORS：RequisitePro对需求从捕获到分析、从分析到跟踪提供了完整的解决方案，具有无以伦出的优秀表现。但是需求管理工具在“雷声大，雨点小”的国内，面临起步阶段的中国软件业，Requisite在产品的易用性、技术支持方面的优势还并不明显。在笔者的试用中，真切地感受到了DOORS在功能设计、产品考虑等方面都对需求管理的新手们更具上手性，竞争仍然还在继续。

● Rose VS Together：Rose伴随着UML一起成长，为分析人员提供了一个最卓越的平台，但由于其一直定位于通用建模工具，而Together从一开始就是与开发工具相结合；加上Together有Borland的广大用户群作为后盾，在吸引开发人员方面略占先机。不过，现在有了Eclipse的Rose，Together将受到更大的挑战。

● ClearCase VS CVS：庞大的ClearCase，也就意味着是复杂的ClearCase。面对配置管理方兴未艾，VSS就能够基本满足需求，口袋还很空的中国软件业，也许CVS这样的开源软件将相对其来说具有相当



的竞争实力。ClearCase LT版是一个开始，那么下一步呢，对国内软件开发业的培育将成为争得更大成功的关键？

这些特点鲜明的单一性产品，也许就像市场上的狼，开始对作为大象的IBM和Rational发起了冲击和挑战，我们将会得到的必然是更大的收获。

顶礼膜拜还是为我所用？

认知Rational，许多人是从对UML的三位创始人Ivar Jacobson、GradyBooch和James Rumbaugh的了解开始的。稚嫩的中国软件业连社会化分工都还没有形成，思想与实践都与欧美软件业有着明显的差距。在笔者的实践与日常的交流中，都发现对Rational，特别是RUP大多是顶礼膜拜，而非为我所用。

这两者之间的鸿沟一方面对中国软件业人士提出了要求，另一方面也给软件CASE工具厂商提供了巨大的市场机会。

结言

自从IBM收购Rational之后，Microsoft、Borland都开始将自己的产品线向软件工程的其它环节延伸，换言之都朝着Rational这种大集成的方向发展。可以预知，在不久的未来，软件开发的自动化、高复用化必将成为现实。是从开发工具延伸到集成的厂商，还是一开始就走集成的Rational会取得成功，我们拭目以待。

整合随需而变的电子商务 Tivoli software 文/牛新庄

在IBM五大软件产品家族中，Tivoli可以帮助客户轻松整合日益庞杂的IT基础架构，并有效控制成本；Tivoli的重点是为客户和合作伙伴提供整合帮助，降低设计、创建、测试、部署和管理这些综合性应用的难度——简而言之，帮助企业进一步迈向随需应变的电子商务。

技术特点

Tivoli软件包含存储管理、网络管理、安全管理和系统管理等功能模块，业界中几乎没有一个软件能够提供所有的这些功能。在网络管理方面，Tivoli的Netview的最大竞争对手是HP的Open-view。Openview在国内网络管理市场占有很大的市场份额，OpenView已经成为网络管理市场的领导者。虽然目前Net-View在吸引第三方应用开发厂商方面还不如OpenView，但这种差距正在缩小。在存储管理方面，Tivoli Storage Manager核心功能包括数据备份和恢复、管理并恢复数据、24×365保护关键性商务，如今在这个市场，它正同CA公司的BrightStor、Legato Net Worker、VERITAS Net Backup等产品进行激烈的竞争。综合比较，CA公司存储管理具有性能优越，功能完善等优点，Net backup在国内中小企业拥有最大的市场份额，但是TSM具有很好的磁带库管理功能。在系统管理（资源监视，控制和优化）方面，Tivoli同样和CA公司有着激烈的竞争。CA公司系统管理产品专业完善，但是高昂的价格也使不少企业望而却步。在安全管理方面，Tivoli安全管理解决方案是一个集成的安全产品系列，能够为集中式的安全管理和访问控制提供综合和可扩展的解决方案。相对于Tivoli的安全管理，CA公司的eTrust产品对安全策略管理支持比较成功并占有很大的市场份额。

所以，Tivoli软件能够提供一整套完整（安全、存储、系统和网络管理等）的企业解决方案，使公司的IT部门用较少的时间作出响应，而将更多的时间用在抓住能为公司带来竞争优势的机遇上。Tivoli的目标是减少客户运行环境中各种流程和子流程的人工和时间成本，从而帮助客户改进自己的基础设施管理。Tivoli软件可使基础构架快速适应企业的变化，并帮助实现IT专业人员管理的内容或系统管理的内容的平衡。Tivoli是IBM为客户创建解决方案而进行的各种工作整合的结合点，当与IBM的“电子蜥蜴”计划（自主计算）相结合时，体现了IBM通过一系列标准在IBM内部以及在行业中推动开发各种产品的能力，在实际为客户提供可管理性的同时，为客户提供更大的价值。

结束语

Tivoli未来需要着重发展目标有四个：保证系统的可用性；系统应用的安全可靠；具有自主优化能力；资源的预置能力。“电子蜥蜴”计划（自主计算）的系统管理也提供了四种服务：自配置，资源可以动态地自我配置；自防护，能够保证提供足够的资源保护；自优化，提高更高的效率；自修复能力，一旦出现故障，可以自动进行修复。总之，依托世界级的IBM服务、支持和研究力量，Tivoli软件必将提供无缝集成的、灵活的随需而变电子商务基础构架管理解决方案。

Lotus software 产品家族漫谈 文/吕翊

上个世纪80年代，Lotus公司就以其犀利的报表产品Lotus 1-2-3闻名于世了，该产品曾一度占据了全球电子表格处理软件绝大部分的市场份额，直到微软以全新的办公套件Office系列特别是Excel等软件全面出击，才使得Lotus不得不面临市场颓势。

1996年，IBM公司正式收购了Lotus公司。Lotus凭借着IBM的注资，成功地

对其基于电子邮件和消息处理的群件系统Lotus Notes进行了进一步的研发，进而推出了服务器端产品Lotus Domino，并且借着当时Internet/Intranet技术的普及而一跃成为全球市场的幸运儿，到目前为止，全球使用Lotus邮件的注册用户超过了1亿8千万，其兴盛可见一斑。

在群件领域，能够与Lotus Domino/Notes匹敌的，只有Microsoft Exchange了。微软的应用虽然具有与Windows操作系统良好的整合性，但是成也萧何，败也萧何，也正是由于Exchange只能运行于Windows操作系统上，极大限制了其在企业级服务器上的运行，而Lotus Domino/Notes则提供了对包括IBM AIX、IBM OS/400、Sun Solaris、Linux、Windows等几乎所有业内操作系统的支持，这就使得它成为了更多高端企业用户的唯一选择。事实也证明了这一点：根据Gartner Group 2004年底公布的最新市场报告显示，Domino/Notes在群件市场占有超过46%的市场份额，位居全球首位。

迄今为止，Lotus Domino/Notes已经发展到了R6.5的版本，而且将于今年第三季度推出最新的7.0版本，我们可以近距离地看看Lotus Domino/Notes的基本功能：电子邮件服务、协作服务、文档型数据库管理、Web应用访问服务。这些功能应该说搭建了企业级协作业务的总体框架，而且就其统一性和协作能力来讲，Lotus Domino/Notes又把这些功能有效地捏合成了一个整体。对于一个普通的IT业内人员来讲，Domino可能是一个比较特殊的平台。因为她采用了文件型的文档数据库格式(.NSF格式)，采用独特的LotusScript语言进行应用设计开发，而且对于一些业务应用采用的实现机理也有所不同。但是对于一个专业的Domino程序员来讲，使用Domino平台，可以设计开发并实现一切他想要实现的应用，因为Domino的确具备了这样的特质。

Lotus Domino/Notes将走向何方？首先，IBM已经将其企业信息门户产品WebSphere Portal Server系列整合到了Lotus产品线上；其次，从2004年起，IBM又推出了完全基于J2EE体系架构的Lotus Workplace产品。被认为“不仅仅是一个复杂的软件包，更是一种思想”的IBM Workplace软件系统，所提供的终端用户工作环境将各种协作工具、应用和连接融合到一个专为满足特殊业务需求而设计的单一环境中。IBM Workplace包括四个主要组件：Lotus Notes/Domino、Lotus Workplace、WebSphere Portal、WebSphere Everyplace，贯穿其中的则是Workplace客户端中间件概念。同时，IBM还将在原有的Notes的基础上不断发展Rich Client技术，进而通过Lotus Workplace实现客户端应用的一统江湖。

个性十足的Lotus　　文/徐宁

Lotus家族体系中的每个族员都很有个性，拥有着丰富的功能。下面简单介绍几款优秀的产品。

产品	功能
Domino	可做邮件服务器、应用服务器使用。提供强大的群集服务。支持J2EE。
Notes	主流产品。二次开发简单，使用的语言多样，如Notes自带的公式、LotusScript、JavaScript、Java、Html和Xml等。具有很强的扩展性。
Workflow	自动化工作流的设计工具。
DominoDoc	分布式文档管理系统，有效的管理文档和信息。
Sametime	强大的实时协作功能。

目前开发办公自动化系统主要有两类产品，它们在市场上有90%以上的份额。

第一类：基于Lotus Domino/Notes平台。这类在中国的用户量相对较大，占有绝对优势。其大受青睐的独特魅力在于：

● Lotus Domino/Notes不同于其他关系数据库系统，它是一种文档数据库系统。善于管理复杂的文挡。它不受字节数的限制，可以存储文本、电子表格、图形、声音、动画等信息。它还具有强大的数据库复制技术，可以实现单向和双向复制，有效同步和备份数据库。

● Lotus Domino/Notes是简单的工作流应用程序。在流程定制方面比较快捷和灵活。如果结合Workflow进行二次开发，可以更好地实现流程自定义。大部分客户所需的工作流都太过人性化，Lotus产品可以很好地解决这个问题。

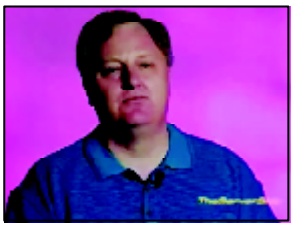
● Lotus Domino/Notes拥有完善的安全保障体系，包括授权/认证机制、数字签名、数据与通讯加密、存取与执行控制、口令保护机制等多种安全保障机制。是目前公认为最好的安全系统。

● Lotus Domino/Notes具有自己的一套邮件路由机制，既安全又方便。在它自身提供的邮件模板基础上只要经过简单的加工，就可以形成客户所需的邮件模板。

第二类：基于微软Exchange平台，这类用户数量也不少。其优点就是能和微软的产品高度集成，缺点是和其他系统的集成较麻烦。

Lotus的下一代产品将以全新的基于J2EE的架构出现，命名为Lotus Workplace。这个新产品是以DB2作为其后台数据库，可弥补Lotus产品中的不足之处，大大提高系统性能。相信它可以给Lotus产品带来一个新的辉煌。

怎样做



实用主义程序员

Dave Thomas和他的合作伙伴Andy Hunt一起开办了“The Pragmatic Programmers”(一家咨询服务公司,专门帮助团队更好地编写软件和提高软件质量)。他撰写了7本书,其中包括《The Pragmatic Programmer》(中文书名:《程序员修炼之道》)和《Programming Ruby》。他的新书《Agile Web Development with Rails》反映了他对特定类型Web开发的Java替代方案的兴趣越来越浓厚。可以通过http://www.pragmaticprogrammer.com与Dave联系。

Q:欢迎Dave,请你向社区的成员自我介绍一下。

Dave:好的!我是Dave Thomas,程序员。我已经从事了近30年的编程工作,听起来可能有些不可思议。最近10年,我一直与Andy Hunt一起工作。我们称自己为注重实效的程序员。我们四处帮助各个团队开发软件,自己也写一些软件,并且也参与了很多流程改进软件的开发工作。最近,我们将收集到的这些信息写进了一本书中,这就是《The Pragmatic Programmer》。当得知在Amazon的排名第九时,我们吓了一跳,真是个不错的消息。在这本书大获成功之后,我们又出了四本书。最近我们开设了自己的出版公司,出版自己的书和其他作者的作品。

Q:你最近推出了入门系列丛书。这背后的初衷是什么?

Dave:初衷非常简单。我们在对所接触的团队和他们所进行的工作进行了研究之后,非常惊讶地发现:为了能编写软件,必须有一系列特定的基本实践操作。这有些像安全装置。假设你必须跨过一根很高的铁丝,而又没有安全装置可用,跨越的速度就会慢很多,并且将尽量少冒险。软件开发也是一样。除非采取了某些基本操作,否则在编写软件时就不能充满信心地开展工作。入门系列书籍就是通过为团队提供他们所需的基本操作,给他们带去信心。我们认为这些基本操作就是版本控制、自动化和单元测试。我们将这三方面分三本小册子单独论述。这三本书都非常直接,非常易于阅读,语言精悍,为团队提供了相关的实践指导,让他们开始慢慢养成这些基本的操作习惯。

入门系列书籍就是通过为团队提供他们所需的基本操作,给他们带去信心。我们认为这些基本操作就是版本控制、自动化和单元测试。

Q:你提到了多样化,作为Java开发人员,应该怎样实现多样化?

Dave:现在大多数人所做的工作越来越集中,人们越来越关注某类特定技术的细节,而这让我们非常担心。我认为Java将会长时间地存在,显然它不会退出历史舞台,但Java是否仍将是服务器端的主流技术呢?我真的不知道。我想有可能会有某种全新的技术替代它的位置。正如我们前面提到的,Java“走红”得非常快,其他技术很有可能会和它一样迅速流行起来,或者可能改变我们做事的方式。例如,我们可能会回到采用更轻量的技术,或者可能会使用Groovy一类的语言将目前很难编码的各个事务项进行连接。我想采用Java也有

一些冒险，就像把所有的鸡蛋都放进一个篮子一样。出于多样化的目的，能在哪些地方可以进行一点小小的变动呢？例如，如果你在开发纯服务器端Java程序，就可以在客户端的Java程序上进行一点变动。越来越多的人开始关注富客户端技术。例如，我们可能看到Applet技术或Java Web Start或其他技术的复兴转而使用Struts或其他方法编写程序。虽然采用了与Java不同的技术，但仍然属于Java阵营内的技术，这样仍然很冒险。为了以防万一，我需要使用更广泛的技术。例如，可以考虑.NET之类的技术。有点像“膝跳反射”，Java社区里很多人都有这种想法：我是Java程序员，而那是Microsoft的技术，我不会去碰它。我认为这样做有些目光短浅之嫌，因为我们面临的环境中互操作性将会越来越强，我们需要和Microsoft前端进行越来越多的交互，不管它会是什么样，我们都需要努力实现互操作性。如果我们需要设置这些标准，并且需要设置实际的标准，那么我们在进行努力，我认为这一点才真正重要。

另外，实现多样化时，我们需要考虑一下非技术多样化。很多开发人员会认为多样化就是学习一种不同的编程语言或学习一个不同的库或者别的什么，但我认为还有其他的事情需要我们去做。我们也需要学习在如何更好沟通方面实现多样化。作为软件开发人员，我们对于清楚地说明我们所创造的价值真的很不在行。外界就看见我们坐在那里，双手敲击着键盘，但一天结束的时候又有谁知道我们所完成的工作呢？我们需要更好地让外界了解这个价值。我们需要多样化地沟通。我们还需要实现行业多样化，进入不同的行业。我来自达拉斯，达拉斯电信部门几年前就基本上破产了，电信部门的人发现工作非常非常难找。如果能找到跨行业或学科的方法，我认为这也非常重要。

此时再次回到最基本的问题上，你所需要的就是如何将自己的时间进行最佳投资，以让自己实现最大的增值。如果你使用了那个甚为简单的方程式，可能会说“这是一个很值得进行多样化的领域，而这个则不是。”想象一下，整个世界在不断改变，而你如何应对这些变化，又能如何定位以处理这些更改呢？假如出现风险最大的情况，Sun倒闭了，这会对我有什么样的影响呢？Java被一种全新的语言取代了，这又会对我有什么样的影响呢？如果考虑了所有这些风险，那么我们所要做的就是通过多样化来防止这些风险给我们带来危害。

Q：你在Ruby脚本语言和其他类似语言方面做了很多工作。你认为它们有什么优势，它们将如何适应企业空间（Java空间）呢？

Dave：这个问题不错。我想这个问题的答案简单地来说就是它们将越来越适应这个空间。为了说明，我想先来回顾一下。首先，Ruby和Groovy、Python或者其他语言都是以不同方式对待编程范型的。它们比很多程序员所习惯的语言更为动态，实际上我对这点很感兴趣，因为这些意味着能提供更多的思维方式。我在这个世界中看到的不是层次结构，也没有看到类图，我看到的是可互操作的对象。有时候互操作性非常松散，用Java这类的语言编程时，我们有时会忽视这一点，而去设置非常严格的结构，例如设定该类是这个类的子类，它具有以下方法……如此等等。我们编程的方式和实际的世界之间有点脱节。在Ruby等语言中，这种脱节不是那么严重，因为这类语言更为动态。例如，没有必要到处都使用继承。在Ruby中，你会发现自己很少使用继承，因为没有必要使用。就我自己的观点而言，Ruby更好地反映了现实世界，因此当我用Ruby编写程序时，我可以不用考虑很多，大脑和键盘间的距离小了很多，并且可以就按照自己想法的产生和发展将之敲出来，这样就能正常工作。有趣的是，很多编写良好的Ruby代码看起来像伪代码，但仍然能执行。我发觉自己使用Ruby一类的语言时效率要高出很多。两年前的夏天，我花大约六个星期的时间编写了一个约35,000行的Web应用程序，并投入使用，没有出现过大问题。如果我使用Java一类的语言，是没有办法完成这项工作的。而且，如果使用Java编写的话，要完成同样的工作，可能需要100,000行Java代码。Ruby让工作十分高效。而另一方面，Ruby采用了不同的编程方法，很多Java程序员都被这给吓到了，他们被告知静态类型化编写的程序更为安全，因此我们认为在Java中每个东西都要有类的类型，如果需要，会进行类型强制转换。而现在Java 1.5中有了这个神奇的东西——泛型框架（一个门户应用程序）。而事实是，如果问问普通的Java程序员，他们最后一次遇到类强制转换异常是什么时候，他们可能告诉你或许是去年吧。这类错误不会出现了，因此在Ruby中也不会遇到此类错误。Ruby没有类型或变量，但并不会导致程序可靠性下降，很少会将类型不对的对象放置在错误的地方。正如刚才所说的，编写IDE时，静态类型化真的很有用。它让我可以完成诸如帮助工具等工作，并完成IDE探究出部件方法的那些项，列举一个所有部件方法的列表，但就编程的可靠性而言，我不甚信服。我认为这些语言安全、高效而且能帮助我们写出更好的代码。就长期而言，这有什么好处呢？我想，或许是Jython或许是Groovy之类的语言将取得中间粘合剂的地位，通过结合生成应用程序，这方面目前的方法可能是使用IOC容器然后借助XML进行连接，而将来可能会使用以Java、Groovy或Jython编写的库，但将通过中央核心连接到一起（此中央核心是一种脚本语言）。这将是一种非常强大的编写应用程

序的方法,应用程序将更具动态性。回到担心类型安全性的问题上,我唯一要说的一点就是:事实上,Java和Ruby一样未类型化。

Q:你为什么说Java未类型化?

Dave:想一想正在运行的Java程序,例如,一个非常小而简单的Swing应用程序可能有20个变量;在应用程序代码中有30个变量;但在运行时仅在堆栈或堆上实例化的就可能会有几千个对象。这些对象都在哪儿呢?你有20个变量,2000个对象。到底怎么回事?很明显,所有这些对象都在容器中,它们都未类型化。在容器中和被使用时,它们都是类型化的对象,我们将把它们取出,然后对其进行强制转换。这和从Ruby中的容器取出对象并使用的过程完全一样,这个对象可以是字符串、用户定义的或任何东西,正如由于在Java容器中的这些对象是正确的,通常不会遇到类强制转换异常一样,在Ruby中也不会遇到此类异常。事实是,有人认为必须采用强类型化以保护我们自己,而我并不认为这无法经得起实际考验。

Q:Ruby如何使你在Java中以不同的方式进行编程?

Dave:问得好。这又要提到系列知识丛书了。与在股票市场投资相比,知识投资中有趣的事情之一就是,如果在知识上进行投资,你几乎不会有什么损失,因为你学到的每样东西都会使你变得比以前更棒。举例来说,两三年前,我们都在学习Haskell以有实效地进行邮件列表编程,而这很大程度上改变了我编程的方式。由于我学习了Haskell,现在思维方式都不同了,Ruby也给我带来了同样的影响。

Ruby的方式很不一般,如果查看一下Ruby的工作方式,你会突然发现其中有些东西非常合理。例如,Ruby中有一个称为块的东西,但并非传统语言中的语句块。Ruby块是变成了对象的一段代码,可以传递给方法,然后该方法就可以从其自身内部调用那段代码了。这是一种功能非常强大的技术。例如,Ruby迭代器就是通过多次调用一个块实现的。例如,数组将调用希望延伸到新位置的块。这是一种非常明显的用法。可能相对不太明显的用法就是可以使用块管理资源和事务等。可能你会使用一个数据库语句,希望能对返回的数据集中的元素进行迭代。而在Ruby中,要实现此操作,可以通过让语句调用一个块,并依次传入返回的每个行,随后块就可以对这些行进行处理。这样的优点在于语句知道何时进行自身关闭。不一定非得将该代码放置在外部,但要将其封装回语句内。与此类似,该语句自身可能实际就位于一个块中,而该块是一个数据库连接,因此你将打开数据库连接,然后将该连接传递进一个块。当该块完成处理时,它将自动关闭连接。在JDBC中编写程序,最后引发了异常时,会遇到很多麻烦的异常处理内容:要关闭什么,必须关闭该语句呢还是要关闭连接,总之很麻烦,而在Ruby中不会遇到任何此类的内容。在Ruby中,数据库程序库代码将对此进行处理。这是一个要好得多的代码编写方法。我现在发现有了Ruby的这种体验后,在编写Java时,我倾向于将更多的代码编写为匿名内部类,这种类基本上和块相同,可以由其他项进行调用,而这意味着我将异常处理从较高的级别移回了库中。而这样做使得代码变得更容易读懂了。尽管在Java中必须采用匿名内部类实现这一点有几分让人脸上无光,但即使这样,对我而言,这仍然是一个非常有效的处理事务的方法。这是在Ruby中编写代码对我Java编程的影响之一。另一个影响得回到敲键盘这件事上来。以前,如果我想将某个复杂的结构从A传递到B,我必须编写某种类,该类会具有全套的getter、setter、实例变量以及其他所有的东西,我在一头对它们进行设置,传递对象,而在另一头得到它们。这样虽然能正常工作,但是却需要敲上半个小时键盘将这个大得有些荒谬的类输入进去。现在,我只需传递一个Hashmap,并将字符串放在其中,因为我认为键盘输入并非真的很重要,并不会带来多大的区别,工作非常有效率。要更改或添加新元素都非常方便。新元素位于其中,如果不需要,就不会使用。我无需更改现有的代码,这样编写代码就非常灵活。我建议那些目前还没有体验过动态语言的人可以尝试一下,看看你能不能克服那种先入为主的想法:“哦,这个不一样,我不喜欢”,继续尝试,直到你习惯它为止。我敢打赌,这将改变你编写Java或C#或别的语言代码的方式。

Q:Groovy正试图将Ruby一类的语言融合到一种JVM语言中。你对此怎么看?

Dave:我想这样很好,这是一种必然的趋势。Groovy或Python之类语言的优势不容忽略,因此这正表明了此类语言时代的到来。对我而言,非常有意义的是Groovy正越来越受欢迎,这种现象很好。我想其大受欢迎的时代正在到来,因为开发人员认识到这种需求,知道自己必须采用一种更好的代码编写方法,或许就是它了。Groovy本身是一个很不错的语言,因为它将很多Ruby中的观念带进了Java世界中。你拥有了一种动态的语言,具有良好的结构,能兼容整个现有的Java库以及你所编写的所有Java代码。这真的很棒,因为我现在能在两个世界中游刃有余。这就是我不需要完全抛弃现有的东西的原因所在。但如果某些东西用Java编写更好的话,我将用Java进行编写,然后使用Groovy等对其进行集

成。Groovy也具有更大的优势，例如，在Groovy中可以进行测试。而这意味着人们将更频繁地进行测试。用Groovy编写测试比用Java编写测试更为简单，可以在测试中更灵活地开展工作，因此人们要进行测试。这对于深入探索非常有意义。如果你希望了解这个库是如何工作的或类是如何工作的，我可以将其放到控制台中进行交互，与它“玩游戏”。我可以插入值，然后看输出结果是什么，而不用编写代码进行此工作。所有这些动态特征都会给人们带来帮助。它一定会使事情变得更为简单。我认为，这些动态语言中的某种方式将成为主流，成为将来我们编写应用程序的方式。是Groovy还是Python，或者某种我们还没有听说过的语言会成为未来的主流，我不知道，但这件事终将会发生。

Q：你认为Groovy需要成为JSR吗？

Dave：这个问题很有意思。不错，Groovy的JSR大约在一个月前启动了。我对此抱着两种态度。我认为存在我们所做的是本末倒置的危险。Groovy仍在发展之中。我们还不知在Groovy处理事务的最佳方法。语言随时都在发生变化。随时都在添加新的特征。我想过早地使其定型化是一个非常大的错误。对于JSR流程我的理解是：JSR将测试和跟踪Groovy 2.0的未来发展，以便二者能够同时对外发行。我个人偏向于这样：推出Groovy，然后让人们试用，随后根据人们的体验对该语言进行调整。一旦基本定型了，然后我们可以说：“好吧，现在让我们将它定型吧。”而我个人认为这才是制定JSR的最佳时机。而另一方面，我怀疑如果没有JSR，如果大家没有说“是的，这是正式的版本，这是正确处理事务的方法”，有些供应商或用户、公司将不会使用Groovy。因此，我认为从这种营销的角度出发，这是很好的做法。但从技术的角度而言，我更希望标准按照实际使用制定，而不是根据某个委员会认为的良好做法将标准强加于人。我尊重Groovy的JSR流程中的所有人，我认为这是一个很优秀的团队，能提供不错的应用程序。不过，我只是担心整个流程可能会有一点“早产”。

Q：最后，你为什么认为软件非常复杂，我们能为此做些什么呢？

Dave：软件复杂是因为世界复杂，软件模拟现实世界。但某种意义上说，软件比现实世界更为复杂。看看周围的世界，复杂性随处得到体现，但我们有很多的后备资源，可以处理出现的错误。在现实世界中，事物从来不按照我们想当然的方式运行，但这不重要，我们糊里糊涂地就想通了所有发生的事情。但问题是，在软件中我们必须模拟这种复杂性，而且还要让其十分完美，而这个任务几乎不可能完成。我认为我们为自己设立了这样的幻想，既要使这个东西非常完美，又要完全模拟现实世界，这给我们带来了很多麻烦。

我所希望看到的是，软件变得越来越简单，而同时变得与现实世界更加接近了。我希望软件充满活力，能更好地处理错误。我也希望在所专注的领域，软件能带来更大的价值，功能却更为简单。因为我们实实在在地为外部世界带来业务价值，而且有比编写庞大的企业Java应用程序更好的处理方式。我认为，如果我们能更好地认识到价值的来源，将最终着手编写更为简单更为高效的软件。我想，在将来，很明显，我们将看到更多的交互代码段，而不是“独行侠”应用程序。我希望看到编程语言和技术不断发展，让我们能够编写此类互相关联的应用程序。我想，这种应用程序的编写会更为简单，我们可能会看到偶然事件，而这会为我们带来更多的价值。最后，软件复杂的原因之一就是当前的软件是由营销部门设计的，而不是技术人员设计的。如果注意一下最近四五个版本的Java运行时的大小等元素，会发现它们实际上是以指数级在增长，而这就是因为我们必须在其中提供各种功能。未来的功能构建于现有功能之上，没有人抛弃任何东西，因而，作为开发人员，你将被所有需要不断学习的这些新内容淹没。你永远都来不及获得对其有效利用的经验，而新的东西又来了，使其变得更为复杂。事实是，我们需要首先考虑开发人员，想想我们要完成这项工作真正需要的是什么？我们需要这些徒增复杂性的库么？我们真的需要部署这个更为复杂的组件才能完成工作么？不错，如果我们在编写亚马逊网站的下一个版本，我们需要适当大小的企业应用程序，但如果我们编写从邮件列表获取电子邮件地址的软件时，我认为我们不需要15个部署说明符和17个Bean来完成这些工作，不过很多开发人员都会陷入进行这样操作的陷阱。如果可以找到编写更为适当的应用程序的方法，忽略那些“遵照我们的最新标准，采用我们最新的库”之类的口号，不管采取别的什么方法，我认为我们的生活都会变得更为简单。

如果可以找到编写更为适当的应用程序的方法，忽略那些“遵照我们的最新标准，采用我们最新的库”之类的口号，不管采取别的什么方法，我认为我们的生活都会变得更为简单。

程序其实也是一种艺术

——袁红岗的程序人生



文/闫辉 贾菡

许多人以为程序员就是每天面对浩如烟海的枯燥数据，进行着机械繁重的工作，毫无想象力可言。可是对于袁红岗这样一位从事了十几年程序开发工作的高级软件工程师来说，写程序其实也是一种艺术创作，需要作家般华丽的想象力、音乐家的激情、哲学家般缜密的思考。在这份他人看来似乎属于纯理性领域的工作里，他总能感受到创作的喜悦和艰辛，如同艺术作品带来的激情和成就感。在这十多年期间，袁红岗一向都很低调，只是在论坛上留下和网友的技术讨论以及自己的经验分享。

“软件是为使用者服务的。”

在大学期间的袁红岗，曾经满足于自己动手制作小软件带来的成就感。一个机会，让在校的他就参与到国家“八五”攻关项目“集成化软件工程开发环境——青鸟系统”的开发研制工作，使用C在UNIX系统之上编写工具软件。在这个项目中，袁红岗主要负责其中“C软件测试工具”的设计与开发。这个项目获得国家科委颁发的“0500系列软件评审”一等奖，北京市优秀软件评比一等奖。这个奖项曾给他带来很大的成就感，但后来他发现，制作出来的软件却被从此搁置了，没有人用它，该软件唯一发挥作用的机会，就仅仅是用来测试了一下其自身而已，这让一度颇感自豪的袁红岗顿时怅然若失。从此他立志让自己写的每一款软件都发挥其应有的作用，而不是让心血凝成的作品被束之高阁。

抱着这一志愿，在1993年毕业之后的袁红岗进入科健公司，参与研制了“动态心电监护——Holler”的软件研制工作，并获得广东省青年高科技博览会金奖，产品投入生产，取得了一定的社会效益和经济效益，此刻他才真正尝到了制作的软件有实际价值的成就感。

在后来的工作实践与探索中，袁红岗对自己提出了更高的要求，在进入金蝶公司以后，他通过学习财务知识，尽量把自己扮演成为用户的角色去了解他们的需求和使用习惯，同时也带领程序员们用换位思考的方式，将自己放到虚拟的使用场景中，体会软件制作中所应注意到的使用合理性。他总是强调，不好用的软件做出来也没有意义，只有让专业财务人员使用起来得心应手、无可挑剔，才称得上实现了软件设计的真正目标。

经过这几次的项目经验，袁红岗逐渐摸索出了自己制作的软件应以“既实用、又好用”的标准来衡量的理念。“软件首先是为使用者服务的。”他认为，技术只是手段，目的是为了实用。袁红岗希望自己做出来的软件有广泛的应用，能真正满足用户业务的需要，符合用户使用习惯，并且有较好的使用体验。

“已经有稳定版本了，我就不要再做，我只做新的东西。”

1994年12月袁红岗加盟金蝶公司时，中国软件在DOS平台上的大型应用软件还非常少，在Windows平台上的软件数量更微乎其微。作为当今中国可以引以为自豪的财务软件在那时还都清一色是DOS版，金蝶的也不例外。但是金蝶的产品已经小有名气，销售前景非常可观，已经给公司带来了很大的利润，完全可以在此基础上稳打稳扎地巩固已有的成绩，继续赢利。但袁红岗不愿就此满足、在前人已经做好的DOS版软件基础



上再重复工作，他更喜欢向新的领域发出挑战，于是在获得公司的支持之后，他开始开发Windows版财务软件的工作。

为了全力支持Windows版财务软件的开发，金蝶将本来盈利可观的DOS开发和销售的工作完全停止，全力投入Windows版财务软件的开发，决心做一次转型的新尝试。在历经一年多的艰苦奋斗后，1997年，金蝶Windows版财务软件终于面世，当年就被《计算机世界》评为中国Windows版财务软件总分第一名，这也使袁红岗一夜成名。

当这个Windows平台的稳定软件版本被开发出来之后，袁红岗再次把工作完全转交出去，开始寻找新的开发领域。原有的金蝶ERP是使用PB进行开发的，功能虽然强大健全，但是没有采用组件化的设计思想，商品化程度很差。他关注到了这一点，开始采用微软最新的DNA体系和C/S三层结构技术来重新开发一套满足分布式计算需要的ERP系统。1999年底，金蝶的招牌产品K3成功面世。袁红岗又在更高一级台阶上获得了新的成就，但是对于他来说，这只意味着旧领域开发告一段落，新的领域又在向他招手，更神秘的世界在吸引着他。不断地向计算机软件技术的更高、更深领域探索已成为他人生的座右铭。

“中间件，我也可以做。”

1999年，J2EE规范发布，成为一种企业级计算的标准，整合了多家传统大型企业级计算厂商围绕JAVA语言开发的各种中间件技术，形成了一套完整的分布式体系架构。在金蝶急于使用中间件的情况下，袁红岗开始了对被BEA收购后的WebLogic进行研究，发现基于Java、基于互联网的应用服务器虽然比BEA的原来的产品好，但同样有不少问题。

当时，虽然有了不少的Java工具，但J2EE平台还没有形成，由于很少在企业中应用，成功案例很少见，很多人都将Java视为一种程序员的业余玩具。但是凭着在国外学习时的阅历和经验，袁红岗果断带领数十位程序员转到了Java，并且认为开发出一个J2EE应用服务器，金蝶不仅能拥有自己的核心技术，还能对这种复杂的分布式计算了如指掌。

在国内，金蝶是最早涉足应用服务器的厂商之一。1998年，BEA在完成收购后迅速向国内推出其应用服务器产品Weblogic；1999年，电子商务的倡导者IBM也推出自己的应用服务器WebSphere。到2000年时，国内市场的应用服务器产品有Sun、Oracle、Sybase和Borland的产品。在同年11月，袁红岗带领的金蝶中间件开发团队推出了拥有自主知识产权，符合J2EE规范的Apusic应用服务器1.0版，产品发布没多久，即获得了第一个用户——天运网的应用。这使金蝶成功地抢滩，占得技术和市场的双重先机。

袁红岗认为大家都在同一个规范下开发、都通过了J2EE 1.4认证，从纯技术角度看，金蝶与IBM和BEA没有质的差别。在软件行业，操作系统、数据库和中间件被称为系统软件的三架马车，国外软件业在操作系统和数据库方面某种程度上已经占据了不可动摇的地位，而中间件在国外发展也没有多长时间，国内完全有可能在这个领域和国外进行抗争。在2005年5月，美国SUN公司正式授予金蝶ApusicV4.0成功通过J2EE最高规范1.4的认证奖牌，并与金蝶中间件签署了战略合作协议。袁红岗命名的Apusic应用服务器是国内第一个通过J2EE认证的中间件产品，迄今为止，全球只有11家同类产品通过该级别认证，在国内的系统软件发展史上写下重要的一笔。

“编写代码只是我十分之一的工作，其余时间都是在思考。”

在常人看来，象袁红岗这样的程序员一定是每天黑白颠倒、象个机器一样日夜埋头编写代码的工作状态。然而事实上在他的开发工作中，先期的绝大部分时间都是在进行思考，如同艺术家的创作一样。他习惯于先在脑子里构思作品的结构与形式，深入挖掘用户的需求，根据业务逻辑考虑软件中可能出现的各种问题，将各个细节和框架都考虑周全之后，再用很少的时间做真正的编码工作。真正动手时一气呵成，整个作品流畅通顺、浑然一体。

袁红岗的住处一度离公司只有5分钟路程，然而他却很少在公司出现，总是习惯在家里工作。要知道，在金蝶公司拥有这样特权的人只有一两个而已，而这也是靠他的成绩获得的信任使然。在开发前期，处于对软件的‘构思’阶段时，他表现得和平常人没有什么区别：吃饭、睡觉、休息、娱乐，思考就与这些正常生活的同步进行。一旦构思完成，就进入了最后冲刺阶段，连续作战，日夜颠倒，在很短的时间内完成产品的编码工作。

“除了编程，我什么都不会。”

袁红岗这样评价自己。的确，一天工作14个小时以上，根本没有时间去顾及别的事情，生活完全由妻子安排和照顾。“她为我放弃的太多了。”提到这一点，他的眼中总是闪过一丝愧欠。

原本兴趣爱好十分广泛的袁红岗，为了软件事业，几乎把原来的爱好都荒废了，唯一剩下的就是对天文学的喜爱。他认为天文学其实和编程有共通之处，都需要非常好的想象力，是对大脑很好的锻炼。面对着浩繁闪烁的星空，他总是能领略到更多更深层次的东西，感受到生命的渺小，也感悟到思想的伟大。

袁红岗觉得，编写软件不仅是他的工作，更是他的兴趣。一个人一生中能遇到一份使他感兴趣的工作并不容易，而兴趣能够成为人一生中全部的工作则是最大的幸运，在兴趣中工作着的人是幸福的。

开放的SAP 拥抱开发者

文/闫辉



SAP 产品与技术事业部总裁　夏嘉曦

夏嘉曦，是SAP产品与技术事业部总裁的名字。不过，你不要误会，他并不是一位华人，而是如假包换的西方世界公民。出生在以色列的他曾创建了多家软件公司，后来被SAP收购后担任了这一职位，他同时也是SAP董事会成员。

就如同SAP全球CEO Henning. Kagermann被翻译成中文名孔翰宁一样，夏嘉曦的名字也是SAP重视中国市场的一个侧面证明。一直以来，SAP被认为只是与全球500强企业联系在一起，但在中国，SAP一直试图告诉人们，中小企业也是它们关注的目标，而不久前由SAP中国研究院开发的SAP Business One产品就是例证。

不过，除了SAP Business One这个地方性策略的产品外，在全球市场上SAP最近有了更大的举动，这便是发布了新一代的技术战略——ESA（Enterprise Service Architecture）和相应的平台产品SAP NetWeaver。

在SAP官方的解释中，ESA被定义为是以服务为导向的架构（SOA）的SAP增强版，即一种将新的灵活性引入IT系统设计的软件架构方式。它依据一系列被广泛采用的标准，明确设定了需要互相协作、共享信息，但却分散独立的不同软件的应用规则。

SAP为什么要推出这样的产品？分析一下就不难发现，这是客户对整合需求的趋势。在原有的R3软件体系中，尽管可以实现SAP和SAP系统间的整合，但现在客户为了保护自己在其它系统上的原有投资，还希望能够将原有的系统和SAP的系统进行整合，这些并不是R3的强项。

SAP北亚区业务推广经理石坚燕女士表示，SAP作为一家德国的软件公司，稳重的作风使得它们没有在互联网热潮的时候放弃原有的技术体系，而仍然坚持了R3的软件体系，而且事实证明这是正确的。但随着互联网泡沫的抹去，Internet技术、Web Service等技术越来越显示出其实在的价值。SAP并非没有看在眼里。于是，一系列的尝试和新产品的开发便开始了。而这便是NetWeaver开发的起因。

同时，我们也可以看到以IBM、BEA为首的中间件软件厂商开始从基础软件向业务层面发展，不断扩大中间件的范畴，当进行到一定程度时，事实上也对SAP这类应用主导型厂商产生了一定的刺激。为了不使自己处于被动的位置，他们从业务层面向下推出平台也是必然的做法。当然，为了更好的保持自己的长处，以业务为出发点是新产品的核心。夏嘉曦说，“随着应用程序和基础结构之间的界限越来越模糊，逐渐形成了这种我们正在定义的‘Applistructure’，未来会出现少量供应商作为业务平台提供者。我们期待SAP能独占鳌头。”

在新产品的架构中，Netweaver包含了两个核心的引擎，一套是以原有R3核心引擎为基础的，另外一套就是Java。NetWeaver除了将原来R3一些非常成功的模块进行重新包装之外，还加入了很多用Java技术编写的新模块。“你可以把ESA看作一组可以独立管理的相关自治单位，但在一般原则和服务级别协议的框架中它们是共同运转的。”

在技术日益标准化的今天，一个产品能够在市场上占据更大份额的做法就是使用更多标准化的技术。石坚燕说，“NetWeaver是一个开放性的平台，更强调面向非SAP的用户。开发者既可以使用这个平台学习SAP原有的管理流程模块，也可以只安装Java的开发平台。”

NetWeaver作为一套向ISV和开发人

员开放的平台，SAP更愿意看到很多系统通过这一平台可以和SAP的系统进行整合，ISV可以在这样的平台上开发一些特定的行业应用，这也可以给SAP的发展带来巨大的好处。当然，SAP也为ISV的开发提供了大量基于SAP所积累的行业流程经验的Web服务和企业服务（Enterprise Service）。

SAP中国研究院院长芮祥麟认为，ESA改变了SAP开发的方式：首先是可以通过由外而内（outside-in）的方法定义产品，编程还未开始，就可以使产品架构展示在用户面前，而且以服务为基础组建应用程序的用户界面。他说，“这是一个真正的开发流程的创新，缩短了产品投放市场的时间，而且这一开发流程同样适用于基于ESA来开发应用的客户以及软件开发商。”

并非只有SAP看到了这一趋势，无论是国内还是国外，都有一些公司已经推出了类似的产品，比如思维加速就一直倡导业务架构平台。还有金蝶软件也推出了类似的EAS平台。但SAP作为企业管理软件市场的老大，这一动作的影响力显得更为重大。

在市场竞争上，思维加速公司CTO宋兴烈认为，NetWeaver更类似于BEA的WorkShop或者IBM的Websphere开发平台。当然它的出发点是从业务角度出发的，而这点又有些类似于金蝶公司最近推出的BOS产品。“当然，NetWeaver是我看到



SAP北亚区业务推广经理　石坚燕



SAP中国研究院院长　芮祥麟

的最完整的以业务为驱动的开发平台。”

由于这一新产品对开放性技术全方位支持，SAP认为这一平台的推出对中国的软件开发企业和开发者来说也意义重大。“假设你是一个开发者或者是小规模的开发商，如何选用平台呢，如果它的应用有世界上最多数量的客户在采用，它具有最丰富业务的收集和最大程度的可伸缩性，如果你想满足这些特质的话，那么NetWeaver就是首选。”夏嘉曦认为，这对所有的中国公司来说都是一个很好的机遇，它不仅使各个公司能够进一步的提高业务运行效率，而且也使中国的软件开发者能够采用一个新的平台，在软件开发方面一步就跨到市场的前沿。“SAP公司投入了数十亿美元投资开发ESA企业服务架构，并且把这样一个架构带到市场上作为一个业务流程的平台。而且这个架构是完全开放的，并不仅限于SAP自身。”

为了支持这一新的技术战略，SAP还开通了SAP研发者网络（SDN https://www.sdn.sap.com/sdn/china.sdn）中文网站，并出版了专门的书籍。在SAP研发者中文网站上，还提供了180天试用版本的NetWeaver开发平台的下载，同时提供大量相关的技术文章和教程。相信对于关注企业软件开发的开发者和ISV来说，这绝对是一个不错的学习环境。

SAP准备以开放的心态拥抱开发者了，无论是基于怎样的市场目的，面对这家企业管理软件领域的巨头，学习这一平台上的开放性技术对于中国的研发人员来说是一个不错的机会，我们也会一直关注SAP技术在中国的推广。

评论：思维加速公司CTO宋兴烈

SAP原来走的是自己的封闭技术体系，这次推出的NetWeaver是对Internet、Web服务开放技术和开放标准的一种靠拢。我相信SAP原来封闭的体系也遇到开放性和应用集成方面的问题。我问了好几个NetWeaver的讲师，问他们SAP今后是否所有R3上的业务都会在NetWeaver上实现，回答是否定的。他们认为NetWeaver是R3在企业应用整合和集成方面的一个很好的补充。比如SAP的人不断强调R3的功能模块可以直接被发布为Webservice，它的Web服务既可以用Java写，也可以用SAP原来的内部语言写，而且也可以暴露为Web服务。

我的感觉是现阶段这个产品主要是针对企业集成需求，并不是要取代R3。但无疑，SAP在应用开发平台和业务基础平台方面迈出了自己的步伐。这也说明，业务基础平台是应用开发的一个必然发展方向。

谈到SAP对中国开发者的影响，我认为最重要的一点就是SAP作为一家关注应用，关注业务的公司现在也深入业务基础平台，说明这是一个不可阻挡的趋势了。另外，NetWeaver基于Internet、SOA、Webservice技术，支持相关的开放标准，说明尽管SOA和Webservice现在还有一定的适用范围，但开发者已不能怀疑其技术不可用了。

从基础技术平台往业务延伸，从应用向基础应用平台延伸是一个必然的趋势，这是市场竞争的必然结果。SAP以前其实内部也有平台，但并没有规模性的推广，也可能是因为与功能捆绑的比较紧密。现在SAP将在市场上作为一个比较新的形象出现，我感觉这是必然的策略。

一个行业软件公司的成长

文/闫辉

在美国硅谷的互联网和软件创业浪潮破灭后，有一大批留美华人开始将目光放到了国内，希望能够将自己的产品或者所学到的知识应用到中国IT领域的发展上。王晖也是其中的一员。他当时已经做到了美国最大的私人高新技术研究所负责计算机软件项目的CTO。


亿网格公司总经理　王晖

王晖的经历颇为复杂，曾经学习物理专业的他研究生毕业后从事过经商，后来感觉自己在商业管理方面还有不足，于是到美国读MBA。然而经过一段时间的学习后，在寻找工作的过程中发现美国公司其实并不愿意让一个外来的、没有西方商务背景的人来做管理工作。于是，王晖立刻调转方向，进入了当时更为火热的IT业，攻读软件工程管理学博士。

毕业后他进入了Backman高新科技研究院，这是由一个美国历史上著名的发明家创办的最大的私人高新科技研究院。他从程序员做起，逐渐担任项目经理，直到最后担当了CTO。离开Backman之前，他负责了一个可以为世界各地的科学家提供完整网格化运算的虚拟空间实验室的项目。

在刚回国的那段日子里，他希望能够将这种"虚拟工作空间"的技术推广给中国的政务工程。这时他遇到了九城集团的老总。九城并非做网游的"第九城市"，而是一家在进出口领域从事IT服务的公司。旗下有很多子公司，亿网格就是一家以开发进出口贸易软件为主的公司。九城的老总看中了王晖的各方面经历，力邀其担任亿网格软件技术有限公司的总经理。

亿网格的创建也颇有戏剧性。公司的创始人也从事过进出口贸易，而当时国内从事进出口贸易的公司在国家政策松动后，迅速从十万家左右成长到三十万家。如此众多的公司每次做进出口业务都需要到国家质检所下面的分支局带着各种不同的文字资料前去报检。不仅费时费力，而且还效率低下。平均一个单子跑下来需要来路上花费大量时间。就在报检大厅中等待的时候，这位创始人忽然想到能不能用电子的手段来解决这些问题。于是找到了几个技术人员探讨，就这样，一项通过Internet进行电子申报的前端软件iDecl产生了，并得到了国家有关部门的认可。

不过，这一软件只是解决了前端的问题：国家质检总部和海关总局都有自己的专网，但普通的企业是不允许直接进入的。如何将进出口企业的报检资料传递给国家质检总部和海关总局进行批复，这是面临的下一步问题。

为此，亿网格公司开发了一套数据交换平台产品。这一平台有两个方面的含义：一方面进出口企业可以在这个平台上通过客户端或者浏览器等前端方式提交报检信息，同时质检总局再从这个系统中提取需要的数据进行审批，这样就将国家机关的内部网和外面信息通过数据交换平台进行了隔离，但可以实现数据的传输。而这一平台也是由亿网格公司开发、维护。客户除了购买客户端软件之外，对其所提供的各种服务还需要交纳服务费用。这对三方来说是共赢的模式。客户节省了时间，国家机关实现了电子化的信息处理，而公司也从中找到了产品和商业模式。

王晖表示，亿网格也从中找到了一条自己的商务模式：这就是以数据交换平台产品为核心，为行业做定制化开发的路子。"如果只推广这种中间件的产品，我们没有IBM这么大的宣传投入和技术推广的支持，客户不愿意采用，但如果只做定制化开发，利润空间又太小。以自主创新的关键产品为核心，提出针对业

务的完善解决方案，既开拓了市场，又能有效的实现利润。”现在，亿网格在产品和项目上的投入基本上按照1:1的比例在进行，人员配置情况也比较平均。王晖说：“这对很多中国软件公司都是最现实的一条生存方式。”

亿网格创办的第一年由于主要是自开发产品，基本上没有什么收入，但第二年就做了近800万的业务。今年，据王晖估计，可以做到一千多万，整个成长的速度令人鼓舞。“我们研发的是可分布部署，集中管理的数据交换平台。我们在北京有行业的中心平台，在广东，上海都有分布部署的分平台。我们的用户量每年要产生上千万份单据甚至更多，都要由交换平台来处理。由于我们的产品是处理海量数据的，一旦抓住了一个行业，那可以在这个行业中比较稳定的发展。”

不过，亿网格并没有只局限在现有领域，随着产品的成熟和市场的稳定。他们发现其实数据交换平台在很多地方都有着巨大的应用前景。尤其是在中国在进入WTO之后，很多管理机构也都需要按照世界标准和规范化的方式进行连通，包括工作效率都需要有大幅度的提升，国家的改革政策也提供了一个广阔的市场。“为各个管理部门之间的整合提供了更大的开放空间，只要按照国际规则和标准化的方法来指导应用，就可能找到流程整合、定制开发的发展空间。”电子政务是亿网格看中的下一个跨行业方向。现在公司承担了北京市市政项目中北京八个区、十三个委办局和相关市属企业的市政管理数据交换的研发任务。

随着业务的深入，王晖发现另外一块市场。原来大家主要做数据方面的整合工作，但在业务流程方面的工作做的很少。比如，某地政府管理部门做了一套管理系统，但发现有解决不了的问题时需要交给上级或其他单位进行处理。然而，相关部门却没有相应的信息系统来处理这些同样流程的事务。王晖说：“这就相当于你坐飞机到了一个地方之后，还需要走很长的土路才能到达目的地。”

如果每次在应用层面整合时，都需要对方也开发一套这样的系统实在是不太现实。看到这样的现状，王晖想到了自己在国外开发的“虚拟协同工作管理空间”的理念。这就是：一个系统的连续性不在于对方有没有系统，而是可以通过邀请对方在自己的系统中做对方需要完成的事情。

事实上，这也是政府机构改革面临的一个重要问题。王晖介绍说，之前机构改革的时候，砍了很多部门，但时隔多年后，很多砍掉的部门又通过各种方式回来了。原因就在于某些事务必须有一定的部门来承担。“比如扶贫工作委员会。这种部门只是一个事务工作，不应该是职能部门。但由于是一个综合性的事务，需要跨部门，而现在没有一个很好的系统、工具或者连通办法来解决这种跨多个职能部门的事务工作。实际上扶贫的工作应该是由财政部牵头，其它的部门来协办的。”如果财政部有这样的一个系统，能够通过工具、通讯办法、文档处理等方式来主持，其它的部门来访问就可以解决这种事务。“点击科技的产品只是注重通讯方面的协同，我们注重的是流程方面的协同。”王晖说。

现在，这套GAI系统已经开发完毕，并得到了很多客户的认同。这也是亿网格在数据整合之外又走出的又一大步，流程整合方面的新的产品和解决方案也成为了亿网格发展的核心。

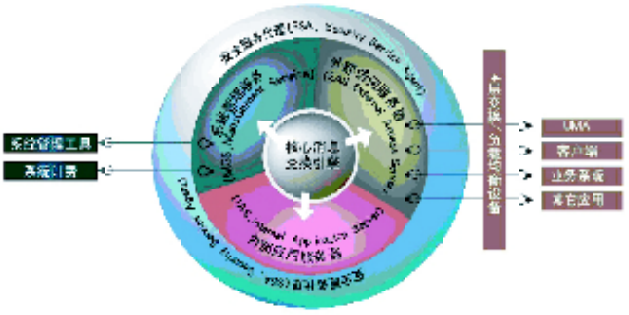
产品准备好了之后，亿网格开始准备在市场上发力了。市场部在不断扩大，王晖说，“包括我自己，现在主要把精力也放在市场上。”

提及这几年管理行业软件公司的体会。王晖认为首先对业务一定要熟，要首先是一个业务专家，然后再去谈IT，谈解决方案。如果没有真正业务上的深入程度，不要去做行业开发。“现在和两三年前，甚至一年前软件开发市场的方式不太一样，以前一直是以技术为指导，互联网泡沫的一个很重要的原因就是技术在引导市场。比如工作流技术其实是程序员设计程序的时候为了把逻辑流程搞清楚而设计出来的软件，而后把这个提炼出来，放到如政府办公系统中去。某些规定的流程可以做，一旦变化就做不了了。所以现在很多OA都没有起到作用。”王晖认为更应该采用成熟技术，不要把“流行”的新兴技术在中国做试验场，也不要把“时髦”技术作为宣传和提高自信心的标志。

同时，王晖认为对流程再造要有创新，有信心。不要担心目前说服甲方很困难，有阻力。“国家其实是下了很大努力的，只要是有效的、创新的，而且管理思路是进步和有科学依据的，就可以坚持创新。”

总之，在行业领域中做开发，要注重长效性。“不能打一枪就走，除了技术准备好之外，还要考虑好商业模式。很多时候甚至要替政府想好有什么商业模式，如何帮助客户。”

亿网格产品架构



声音&幽默



利用数以百万人免费下载的软件赚钱，这可能吗？WinZip Computing的新主人希望能够做到这一点。

——以开发压缩软件闻名的WinZip Computing已经被Vector Capital收购。Vector已经证明一直不赚钱的软件厂商能够赚到钱。它在2003年收购了一直亏损的Corel，并进行了大幅度重组。Corel现在已经赚钱了，它的盈利能力非常高。

我的目标是让每一位用户都接触并使用上最先进的技术，同时我也想为中国的发展和创新贡献自己的一份力量。加盟Google使我有机会实现上述两个目标，我现在非常期望返回中国，开始我新的工作生涯。

——微软副总裁李开复闪电加盟Google任中国区总裁，引起轩然大波。

与Bill Gates会面，是仅次于参观迪斯尼乐园的重要体验。

——来自巴基斯坦木尔坦市的小女孩Arfa Karim是该国历史上最年轻的微软认证专家（MCP），也是世界最年轻的微软认证专家之一，在上周与Bill Gates会面后如是说。

这个转换过程将非常简洁，与其他转换过程相比，这种转换的速度要快的多。

——比尔·盖茨承诺"64位时代"是非常伟大的。从8位计算转换到16位计算时，要求重新编写操作系统，甚至在转移到32位计算十年之后，仍然存在许多问题让人感到有些杂乱。但是根据微软老板的意思，转换到64位的硬件和软件时，一切都将与原来有所不同。

OneCare标志着微软进入了梦寐以求的反病毒领域，挑战赛门铁克、McAfee、趋势等市场霸主。

——OneCare集反间谍件软件、反病毒软件、防火墙、Windows PC调整工具于一体。尽管价格还没有公布，但微软曾表示，正式版本的OneCare将是基于订购模式的服务。OneCare面向消费者用户而非企业用户。

根据我们的研究，今后的家用网络速度最少需要30Mb/s，现在光是电视节目就需要占用10到20Mb/s的带宽。所以，我们很快就会需要30M以及50M的网络。

——一家芬兰公司周三宣布，最早在明年，通过有线电视线路接入宽带网络速度将达到100Mb/s，这比目前一般的有线电视网络速度要快上50倍。

我们做出这样的决定是经过相当长时间考虑的，在过去9年中，我们都没有推出新的OS/2操作系统版本了

——IBM在今年年底的时候将停止销售OS/2操作系统，此举标志着这位蓝色巨人一个划时代标志产品的结束。对此，IBM发言人Steve Eisenstadt说。

要部署一种娱乐软件需要9个月的时间，而手机在市场上的寿命也只有9个月。

——在手机上，开发商必须修改它们的软件，满足不同手机的要求。开发一种软件的多种版本意味着增加数个月的工作量。NM负责运营商营销的副总裁克雷格对此深表担忧。

我们在Office 12中开发的创新技术——新的文件格式、新表单能力、尚未公开的其它新技术，将向合作伙伴提供巨大的机遇。我们估计，Office 12平台每年将带来1400亿美元的合作伙伴服务。

——在为期三天的合作伙伴会议上，鲍尔默大力向6000多家应用软件提供商、硬件转销商、经销商、系统集成商、软件开发商推销微软的技术方案。

安全增强、新的搜索机制、大量新笔记本电脑功能、父母控制、更好的家庭网络。借助Avalon，Vista在视觉方面也有所变化，包括从透明窗口到代表文档本身的图标。广泛的IPv6支持、改进的服务器存储数据在客户端上的缓冲、整个卷的加密、修正过的同步引擎、支持配置有第二个显示器的笔记本电脑的能力、自动硬盘优化、PC丢失或被窃时有助于其他人访问用户数据的安全启动过程。

——新版Windows终于有了"大名"：Windows Vista，微软还发表了这次操作系统的新的功能列表。

Sun向AMD提出了一些使Opteron芯片更适合应用在工作站中的技术性建议。

——Sun可伸缩集团的执行副总裁大卫表示，两家公司间的协作将对AMD的芯片产生协力效应。目前，Sun与AMD在其它技术项目上进行了合作。

未来5-10年内，开放源代码软件将改变软件产业。

——在斯坦福大学的"AO 2005创新峰会"上，数名顶级业界官员称软件产业正在构筑一个全新的世界。

在Harmony项目的方向和目标更清晰前，IBM选择了等待。我们真的希望看到这一社区开始"启动"，我们不希望破坏这一过程。我们希望看到开放源代码版Java。

——据IBM的一名官员称，IBM已经开始加入开放源代码项目Harmony，并计划向该项目捐赠代码。

走出软件项目的死亡之旅

—— 软件项目的管理现状和出路

■ 专题策划：CSDN管理频道

今天，虽然在采用了系统化的软件项目管理方法和质量保证体系后，许多软件开发组织的软件项目成功率确实有了一定的提高，然而越来越多的软件项目往往不得不接受通常情况下被视为不切实际的限制条件：人手短缺、资源匮乏、资金不足、工期紧张。这时，项目经理仅仅利用软件项目管理的常规方法已经不能再保证项目的成功。

为了了解用户在软件开发项目中项目经理遇到的各种问题、使用的各种项目管理方法、项目团队的组织结构等，6月以来，CSDN管理频道对近600名软件企业的项目经理、技术总监、SEPG Leader、软件开发部门经理、质量管理部门经理进行了线上的调查和线下的访谈。通过调查，我们了解了目前软件开发项目的管理现状。针对这些现状和问题，我们对现代卓越集团副总裁兼上海卓而越管理技术咨询有限公司总经理周浩宇先生进行了专访，请他对我们调查的结果进行了深入的分析和点评。

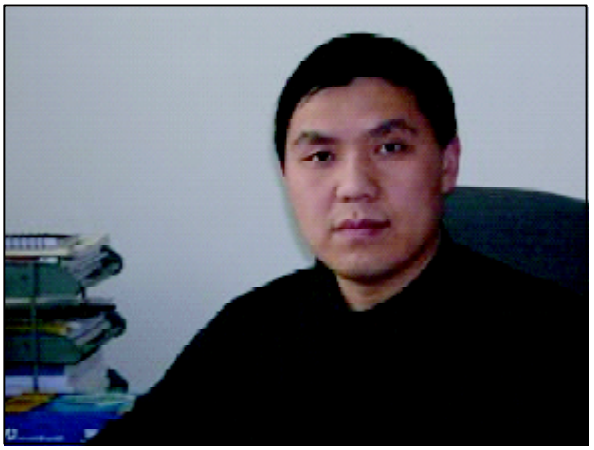
关于软件项目的规模

1.您目前项目组开发人员的数量？

2.公司单个项目规模一般是多少人月？

项目管理人员必须尽快走出自己的小圈子，通过沟通交流提升自身的管理效率。

我们设置了两个问题：项目组开发人员的数量、单个项目的人月规模。调查结果显示：参加调查的60%项目组的规模在1-10人左右，80%项目组的规模小于30人。少数项目超过100人。而从人月的角度考虑，5-12人月的项目和5人月以下的项目占70%左右。这与去年的调查结果相似，目前软件项目规模都还偏小。当然，这样的结果也同时与我们采集的样本有关，大多数参与调查的样本，来自中等规模或规模较小的企业；但的确很多时候，规模较大的项目，会采用项目群组管理或者将项目进行分解，而分解后的项目规模则相对要小很多。



《程序员》:您是如何看待目前软件项目的规模都偏小这样的现状的？

周浩宇：根据多年以来的项目经验，我发现中大规模软件项目的项目经理、项目总监通常都忙于处理项目中的各种事务，往往没有时间去进行各种各样的沟通交流。之所以出现这样的现象，部分是因为项目中缺乏行之有效的流程方法，部分也是因为这些项目管理人员负担着较大的责任，压力巨大，没有精力和时间参与各种交流沟通活动。当然，很多大规模项目的项目经理沟通意识薄弱也是一个主要因素。调查结果也基本证明了这一点，虽然调查样本面向了所有规模的软件团队，但积极参与调查的往往是那些小规模团队的项目成员或中大规模软件项目中处于管理较低层面的项目开发人员，而那些处于高层的项目管理人员参与度相对较低。

而从另外一个角度，并不是说目前的软件项目规模小。在这些年的咨询过程之中，我接触了形形色色的企业，同大量的项目管理人员进行了深入的沟通和合作。在我所接触的这些项目管理人员之中，有相当一部分都在管理规模超过100个人月的项目。然而，这部分人员往往迫于巨大的市场压力，每天疲于应付层出不穷的项目问题，虽然很多人有心参与各种管理交流活动，但却缺乏相应的时间和精力。

我对项目经理的建议是：中国的软件事业要发展，项目管理人员必须尽快走出

自己的小圈子，通过沟通交流提升自身管理效率。而对于高管，需要努力创造条件让他们跳出解决琐碎技术问题的怪圈。

关于软件项目经理和项目管理专业人士

软件项目管理与其他企业的项目管理有很大的差别。软件项目经理将不仅仅负责项目的管理，还要对整个软件产品负责。软件项目经理将负责分配资源，确定优先级，协调与客户和用户之间的沟通。

《程序员》：请简单介绍项目管理知识体系由来，及这几年在中国的发展。

周浩宇：项目管理知识体系是国际上各个项目管理组织根据大量项目实践总结出来的项目管理知识框架。国际发达国家在很早就开始采用这些框架，而且依靠它逐步提升了自身的项目管理能力，大大增加了项目的成功率。在中国，项目管理知识体系的引入始于1999年，从现代卓越1999年同PMI进行合作开始引入PMP认证到目前，项目管理知识体系已经被越来越多的企业所采用。对于项目密集的软件企业，项目管理知识体系也越来越受到重视。

美国PMI的体系具有简单、灵活、易于操作的特点，因此在全球范围内得到了最广泛的推崇。虽然拿到PMP认证并不意味着能够完美地管好项目，但客观上PMP认证起到了推动个人项目管理能力发展、建立统一的企业项目管理平台、提升项目管理专业化的作用。

3.有多少员工通过了PMP或IPMP认证？

谈到在软件行业项目经理的专业化程度，我们的调查结果并不乐观。首先，从项目管理知识体系的引入来看，PMP考试从2000年引进到中国，已经有了5个年头。从最开始的很少有人知道，或被那些爱开玩笑的人士笑称为“拍马屁”考试，到现在越来越多周围的人开始关注这项认证，开始参加这项考试。可以看出，专业化的项目管理知识、以及专业化的项目经理人士，已经在越来越多的被引进到软件开发项目中。从这个调查数据可以看出，目前近3/4的软件企业仍然欠缺专业的项目管理专家（PMP）。

《程序员》：软件企业里拥有PMP认证的比例还很低，您对我们的调查结果如何看待？对那些积极准备PMP考试的开发人员，有什么样的建议？

周浩宇：与国际相比，目前我国软件企业中的PMP认证比例还处于相对较低的水平。但越来越多的软件企业都开始认识到专业项目管理能力的重要性。软件项目的工程化客观上也推动了这一趋势。目前很多企业都在推行CMM、CMMI等体系，而这些体系无一例外地都把项目管理作为最重要的组成部分之一，因此软件企业中越来越提倡通过PMP认证。广大的软件项目从业者也认识到通过PMP认证会为自身带来更广阔的发展前景，因此努力拿到PMP已经成为很多软件企业中的时尚。

对于PMP认证备考，市面上已经存在很多参考文档。我只提醒备考人员一点：PMP考试准备关键在于理解，千万不要死记硬背。理解了国际上规范的管理思路之后，即便题目中的案例背景千变万化，你都可以快速找到正确的结论。

4.您认为项目经理最重要的素质：

对于项目管理人员应该具备的基本素质这样的问题，我们的调查结果显示：“团队管理和领导能力”、“沟通和人际交往能力”，是项目经理最需要具备、也是那些从技术走向管理岗位的项目经理最需要提高的能力。

《程序员》：您如何评价这一结果？您认为软件项目经理最应具备哪些素质？

周浩宇：软件项目充满了各种各样的变化，能否让项目相关人员时刻团结在自己的周围，自始至终地为项目提供支持直接决定了项目是否能够成功。要达到这一点，软技能起着不可忽视的作用。然而，在多年的项目管理培训、咨询过程之中，我发现软件项目管理人员往往更加关注自己所习惯的技术问题，对人员问题视而不见，有时甚至故意躲避。究其原因，这很大程度是因为软件项目管理人员往往非常欠缺软技能方面的规范训练。对于软件企业，能否让自身的项目管理人员全面发展，培养出大量能“软”能“硬”的项目经理，将直接决定企业是否能健康持续地发展。在个人层面，我建议软件项目人员应该有意识地培养自己的软技能，只要你拥有这样的意识并积极尝试，你很可能就会走在前面，必将有利于自身的职业发展。

关于软件项目的组织和管理

5.企业的组织结构是：

首先，我们对软件企业的组织结构进行了调查，我们将软件企业的组织结构分为以下4种类型：项目型、职能型、矩阵型、混合型。调查结果显示：项目型组织和混合型组织各占1/3强，二者加起来占70%以上。

《程序员》：您是如何就这一调查结果进行分析的？

周浩宇：与其他行业相比，这说明软件企业非常重视项目管理，从组织结构上为项目管理的实施作了大量准备。在这样的组织结构中，项目经理往往具有更多的自主权。

6.项目经理的主要来源是？

“从技术到管理”，是很多开发人员的

职业发展路线。对于“项目经理的主要来源”这个调查问题，我们得到的结果是：一半以上的软件项目经理，来源于技术专家。他们在技术熟练到一定的程度，对项目的了解到了一定的程度，就逐渐开始带团队，逐渐开始走向管理岗位。这也是我们在线下的访谈中获得的普遍答案。

《程序员》：很多软件企业的管理人员都是走的“从技术到管理”这一条发展道路。您认为导致这一结果的主要原因是什么？您对这些人有何建议？

周浩宇：这的确是目前存在的一个普遍现象。就这一现象进行分析，我认为，这个过程有时并不仅仅是开发人员自发完成的，而是由高层管理者的行政命令所导致。国内的项目管理目前还处于起步阶段，大多数软件企业里并没有形成规范的项目管理氛围，技术出色的项目人员往往能够获得较多的认同，因此在很多软件组织中都是想当然的项目经理人选。此外，高层管理者直接将技术骨干任命为项目经理还有另外一个原因：在目前竞争残酷、压力巨大的项目环境之中，从技术岗位走上来的项目经理往往能在第一时间内组织资源解决掉面临的技术问题，即便组织不到资源，项目经理也可以依靠自身技术能力亲自找寻问题解决方案。在从技术走向项目管理的过程中，最容易出现的问题就是对技术问题过于关注，往往会忽视项目的整体控制，忽视项目同企业战略的相互关系。

因此，对于企业而言，要把技术专家发展成为优秀的项目管理人员，一定要注意培养这些技术专家的管理能力，让他们尽快放宽眼界，从各个不同的角度看待项目，力争成为全面发展的管理人员。而对于个人，我建议大家应该有意识地培养自己相对薄弱的各种管理思路和软技能。在中国，从技术走向管理的趋势必将持续下去。

7.项目经理一般有什么样的权力？

8.项目经理职权和部门经理相比如何？

接下来我们又对“项目经理一般有什么样的权力”，以及“项目经理的职权和部门经理相比，权力更大还是更小”这样两个问题进行了调查。调查显示，在众多项目经理的职责中，“组建项目团队”、“项目计划、进度监督和质量控制”、以及“技术方案确定”，是最主要职责和权力，各占1/4左右。而近1/2的人认为，项目经理和职权要小于部门经理的职权。

《程序员》：您认为出现这样的局面的主要原因是什么？

周浩宇：与几年前相比，目前的大多数软件企业都已经能够比较专业地看待项目管理所需的权力，也能够赋予项目经理相应的权力，这是一个好的趋势。在调查结果中，技术方案确定所占比例颇大说明项目经理与技术有脱不开的关系。对此，我仅仅提示一点：项目经理一定要摆正管理与技术之间的关系，不要过多干涉技术专家们的工作，切记自己的主要工作是项目的总体控制和把握。

在软件企业当中，虽然组织结构似乎变成了有利于项目管理的类型，但很多情况下往往是“换汤不换药”，仅仅是一种形式。从我所接触的大量软件项目经理的反馈来看，“授权不足”往往是最大的抱怨之一。对于软件企业，要想提升自身的项目成功率，一定要重视这个问题，及早解决，否则必将带来巨大的内耗。对于软件项目管理人员而言，这种现状也意味着软技能、影响力等方面的能力在现在至关重要。要想做好项目，自己必须在这些方面进行较大的投入。

在从技术走向项目管理的过程中，最容易出现的问题就是对技术问题过于关注，往往会忽视项目的整体控制，忽视项目同企业战略的相互关系。

关于项目管理过程

9.您管理的项目生命周期是属于那一种模型？

我们首先对项目生命周期的类型进行了调查。在这项调查中，我们把软件项目的生命周期分为以下5种类型：

- A：编码修正模型：未成型的系统规范－>不断的编码修正—>发布
- B：渐进模型：最初概念－>设计和实施最初原型－>不断的精化原型直到可以被接受(开发一个版本 ->交付该版本－>得到用户反馈－>并入用户反馈)－>完成和交付最终版本
- C：瀑布模型：软件概念<－>需求分析<－>架构设计<－>详细设计<－>编码和调试<－>系统测试
- D：螺旋模型：确定目标，备选方案，约束条件－>识别和解决风险－>评价备选方案－>计划下一个迭代－>交付下一迭代解决方法－>确定目标，备选方案，约束条件
- E：没有模型，完全根据实际情况进行调整

调查显示：渐进模型和瀑布模型在软件开发项目生命周期中比较普遍，占1/2强。这两种模型也是被证明比较有效的两种模式，在开发带有极大成长性的系统、提供给管理者和用户可视的项目进展情况、管理成本的可控性、管理中途变更、以及开发高可靠性的系统方面，有着比较强的能力。

另外有约40%的项目团队选择编码修正模型或没有模型，这两种状况更多的被证明是：没有充分理解客户的需求、没有充分的理解架构、管理风险差、管理成本高、不能提供给用户和管理者可视

的项目进展情况。

和编码修正模型相对的是非常成熟的螺旋模型。螺旋模型是一种以风险为导向的生命周期模型，它把一个软件项目分解成一个个小项目，每个小项目都标识一个或多个主要风险因素，直到所有主要风险因素都被确认。

几种生命周期模型各有差异，在具体的运作中，要根据不同的项目有不同的需求，最有效的模型完全根据项目需求。如果滥用的话，好的模型也会表现的很差。

10.项目团队的工作中，经常会遇到一些外界的干扰，干扰的来源主要来自？

在接下来的调查中，我们了解到，在项目团队的工作中，主要的干扰来自客户。有60%以上的被调查者选择了客户。而来自高层领导的干扰也不可忽视，占25%左右。

《程序员》：为什么客户和高层领导会经常的对项目工作产生干扰呢？而项目经理如何面对这些干扰，您有什么样的建议？

周浩宇：软件项目最大的干扰源来自客户，这说明目前软件项目管理最令人头疼的方面就是需求等与客户相关的变更，因此如何关注客户，与客户良好协作，尽快发现变更的趋势，减少不必要的变更就成了软件成功的关键因素。要想做到这一点，项目管理人员必须发展自身的沟通技能，与客户形成良好的协作关系。当然，除此之外，项目管理人员还需要时刻做好变更的准备（需求双向追踪、良好的扩展性等）。

除了客户，高层领导是另一个巨大的干扰源。对此项目管理人员应该认识到项目最终的目的是实现企业战略，而高层领导的所谓“干扰”往往都是源于市场、商务层面的变化。因此，软件项目管理人员应该认清自身项目同企业战略的关系，弄清高层领导的目的，也只有这样，项目管理人员才有可能逐步提升自身，走向更高的职业发展层面。

当然这个统计结果也警告了企业高层领导和项目客户，“团队将自身看成了导致项目失败的主要干扰源”，这显然并不是他们愿意看到的结果。为了项目的成功，领导和客户必须保持与团队的沟通畅通，及时告知团队自身的意图，这样才能真正实现项目的辉煌成功。

11.公司最常使用的项目管理及开发管理的软件工具是：

谈到工具，调查显示，最常用的项目管理工具是MS Excel，与使用其他开源工具的用户各占1/3左右，其次是MS Project。使用Rational开发工具(Rose/Reqpro/Clearcase/Robot/)的也占有一定的比例。

《程序员》：您如何看待工具？

周浩宇：要想提升效率，工具是至关重要的一个环节。与其他工具不同，项目管理工具是一种特殊的工具，你必须掌握项目管理的相应方法才能正确高效地加以采用。从Excel被选作项目管理的最主要工具可以看出，大多数软件企业还没有掌握系统规范的项目管理方法，因此还局限于静态计划的项目管理层面。至于其他开发工具的入选，则说明了很多软件项目管理人员把软件工程的专用工具与项目管理工具混为一谈。对此，我建议企业应该结合项目管理方法的推广、培训选择适合自身的项目管理工具，要想走向卓越，这将是必不可少的一步。

12.在项目管理应用过程中，经常会遇到一些实际困难。以下哪些符合您公司项目的情况？

针对这个问题，我们一共列举出了15项项目经理经常遇到的问题，从调查结果来看，没有一个问题“落选”，它们受关注的程度也呈均匀分布状态。但相比较而言，以下几个问题更加突出一些：(1)时间进度总是跟不上；(2)质量不能另人满意；(3)项目成员积极性不高；(4)项目范围不断变更；(5)项目选择缺乏规范管理。

《程序员》：为什么会出现这样的情况？软件项目真的是问题重重吗？

周浩宇：现在的项目往往存在一个恶性循环“进度压力大->放弃质量->返工大->成员士气低->积极性不高->放弃规范的管理->进度压力更大”。而变更等情况更加恶化了这个循环。究其原因，主要在于项目管理人员不能保持正确的思路：质量是不能放弃的，它直接决定了用户的满意度。进度如果不能调和，项目管理人员应该坚持自己的原则，划分相应的优先级，首先实现能够保证客户满意度的需求或功能，选择增量开发、渐进交付等生命周期，同时加大同客户、领导的沟通，相信会有较好的结果。限于篇幅，具体的思路不再赘述。

13.您认为以下哪些是导致项目失败（死亡之旅）的主要原因：

我们一共列举了8项导致项目失败的原因，其中受关注程度最高的一项是：“市场人员、高级管理人员、缺乏经验的项目经理所做出的幼稚承诺”，占1/3强。其次是“年轻开发人员的乐观主义：“我们周末能够完成！”，和“出乎意料的危机”，各占20%左右。需要指出的是：“公司政治斗争，导致开发人员和项目经理在选择立场方面出现了矛盾”、“市场竞争导致的：最先将产品投入市场，同时削减成本”，也是重要的原因。

《程序员》：您认为导致项目失败的最主要原因是什么？

项目计划是项目成功的基础，而沟通是所有管理措施实施的手段和准备。控制是计划的目的，而客户则决定了项目是否成功。

其他原因都需要项目管理人员用规范的思路加以规范，传授给项目人员如何做项目的正确原则。

周浩宇：虽然很多幼稚的承诺确实来源于市场人员、高管、项目经理的经验缺乏，但不容否认的是很多所谓幼稚的承诺都来源于市场压力，虽然技术上看来是幼稚的，但市场上看来却是非常现实的目标。对此项原因，高管、项目经理应该尽量与团队沟通，让他们弄清项目同企业发展的关系，另一方面他们还需要采用规范的方法来管理项目，不要让市场压力破坏了自己的做事原则，相信这样操作之后，大家能更好地达成共识。

极度乐观需要项目管理人员给予更多的引导和监控，建立一种健康的项目氛围。

14.您认为以下哪些是项目成功的最关键因素？

在这项调查中，我们一共列举了7项关键因素，选择最多的几项依次是：(1) 有一个考虑周全的项目计划；(2) 有序、有效的沟通渠道，从而防止问题的产生，或者在问题产生时使问题对项目目标产生的负面影响最小化；(3) 及时、定期的监督实际的项目进程，并和计划做比较，如果有必要，立即采取措施纠正；(4) 与客户及时充分沟通，让客户积极参与到整个项目中来。

《程序员》：您认为项目成功的最关键因素是什么？

周浩宇：调查结果也证明了我多年咨询培训所得到的结论。项目计划是项目成功的基础，而沟通是所有管理措施实施的手段和准备。控制是计划的目的，而客户则决定了项目是否成功。相信在CSDN各个渠道的大力推广下，中国的软件项目必将做得越来越好，我也祝中国的软件行业的发展速度越来越快，早日加入世界的顶尖行列。

作者简介：
周浩宇，现代卓越集团副总裁兼上海卓而越管理技术咨询有限公司总经理。毕业于北京邮电大学，中国早期认证PMP之一，中加工商管理学院、北京大学MBA客座教授，编译《死亡之旅》、《咨询的奥秘》等系列丛书。曾就职于西门子、惠普、安捷伦科技等多家大型跨国公司，具备10年以上大型项目的实际管理经验。

更多关于项目管理专题的信息，请访问：
www.csdn.net/subject/rpm

软件项目成功的要素

—— 软件项目的管理现状和出路

文 / 王晓毅

曾经有个笑话，说三个软件高级人材等待上帝安排工作，一个说自己擅长抽象思维，上帝说那就做系统分析师吧；一个说自己工作非常细心，上帝说那就做QA；最后一个说，我实在没有更多的才能，那就做项目经理吧。有句项目管理名言则是这个笑话的最好解释：对项目经理的知识要求是要有1英里宽，7英寸深。也就是说，各方面的综合能力是项目经理的首要技能。

项目管理引入中国好多年了，除了国外的PMP、IPMP认证体系，现在更是将之引入高等学位教育。除了最先应用项目管理的建筑行业，现在各行各业都非常重视行业内的项目管理，这些足以可以看到项目管理的蓬勃发展。但是软件项目失败案例还是比比皆是的今天，如何将项目管理与新理论和技术层出不穷的软件产业双剑合璧？项目管理理论是欧美国家伴随着生产管理起步的，虽然方法论是通用的，但是如何在软件开发中产生更大效益，需要更多的业界项目经理以及高层思索和总结。

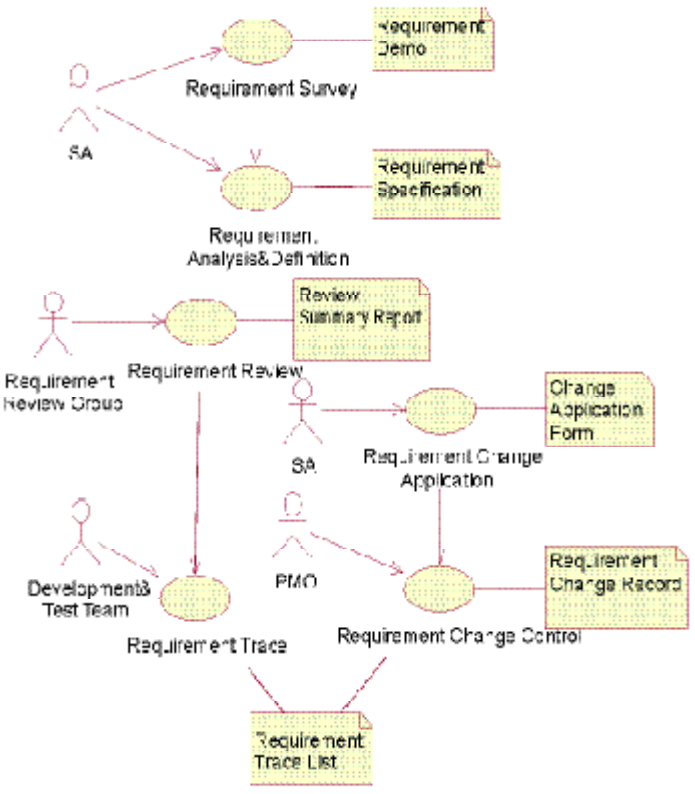
一个成功的建筑行业项目经理也会是一个合格的IT项目经理吗？项目管理有之一个名言：一个成功的建筑行业项目经理也会是一个合格的IT 项目经理。

在欧美国家是适用的，这样跨行业的例子也非常多。但据我了解在大陆这样的例子还非常鲜见。尤其软件开发行业，就更没这种先例了，为什么在欧美或者印度模式中，都是行得通，在中国不行呢？欧美或者印度模式的项目经理负责制定开发计划、协调、以及填写各种项目输出表格或模版就够了。在这种模式中项目经理不一定要求必须是技术专家，但更强调项目经理的工作经验，一般会要求项目经理有8年以上工作经验。而在中国，尤其是现在一个有三年工作经验的team leader也会称项目经理，当然他也并不需要对成本、人员、采购等众多项目管理领域的关注，这样的team里也根本不会有技术经理或顾问专有角色的配备，项目经理要更多的关注技术因素，所以项目经理一般是与行业和产品同步成长起来的。如何做好质量、成本、沟通、时间、以及更资源参与的全面项目管理是我们的项目经理的课题。

范围管理是项目经理具备的首要能力。范围说明是未来项目决策的基线，也是衡量项目是否成功的标准。在我们软件开发项目中，需求规格就是我们项目的范围的体现。我的经验是项目经理应极端重视需求，成功的需求管理才能保证范围在基线内，需求调查、讨论、分析、归档、review、变更、回溯等一系列活动是我们需求管理的有效活动。举例说明：右图是我负责的一个项目，需求开发和管理的流程图。

计划能力是项目经理的应具备的另一个技能，其中软件开发任务的估算是一个难点，即使有历史数据达到CMM4的软件企业也会有20%-50%的误差，我的一些做法用三层到四层的WBS模版从底向上进行时间资源的估计，会从自己经验和相似项目的历史数据中进行加权平均，时间资源的平衡，在WBS分解模版中采用自底向上估计，估计时我们采用了三人以上匿名delphi法，设定差值阈值为30%，如果与平均值的差值比小于此阈值，将不在重新估计，如果大于将进行重新估计，重新估计后如果还是超过设定阈值，估计人要写明为什么如此估计的原因。对于阈值内估计值我们采用历史经验数据进行修正即可作为我们WBS工作包的估计值。

合适的软件开发生命周期模式，对软件开发项目尤为关键。根据项目的需求、资源、风险、时间、质量等实际情况，选择合适的软件开发生命周期模式，对软件开发项目尤为关键。印度软件模式中更是提出了流程模式重于项目。在需求不确定、变化较频繁的项目我们可以选用迭代和原型法。在产品按版本递增开发的项目，由于每期需求比较稳定，宜选择瀑布变种的V模型进行测试提前的生命周期。开发模式的选择将影响项目计划，例如V字形，每个过程都有严格的输入输出，上一个过程的输出作为此过程的输入，实际情况中我们会选用改良的V模型，一些过程可以并行，例如需求规格完成后，可以系统测试计划和概要设计并行。在实际项目管理中开发模型生命周期中各过程的输出宜作为mile-stone(里程碑)设置计划控制点。



时间、质量和成本是衡量项目成功的三要素，时间和成本因为有形比较容易监控，质量控制在软件开发项目中非常重要了，所以我们很多过程和活动(文档review、代码走读、单元测试、集成测试、系统测试、以及各过程的需求反馈追踪等)都是保障质量的活动。现在好多项目都特别重视了系统测试（包括功能测试、性能测试等），但是忽略了单元测试。单元测试是所有测试中最底层的一类测试，是第一个环节，也是最重要的一个环节；是唯一一次有保证能够代码覆盖率达到100%的测试，是整个软件测试过程的基础和前提；单元测试防止了开发的后期因BUG过多而失控；单元测试的性价比是最好的。在项目中我们引入了TDD单元测试实践，并关注了编译检查、代码走读，以及在等价类、边界值、因果图等方法下的黑盒功能测试和达到覆盖率指标下自动单元测试代码的白盒测试。并将XP方法中的Nightly Test实践达到代码和测试代码的每日building。

Review活动在我们的项目中重视度很高，评审首先进行小组内部预评审，内部评审模版记录提交项目经理和评审组织者；正式评审可以先走email评审，评审对象完成人将评审文档或其他可交付物提前一天email给所有评审人，请评审人各自将意见email给评审组织者，评审组织者按评审人员汇总整理意见，提交第二天讨论。讨论后，评审结果记录和每人评审意见列入文档考核绩效。

软件开发项目中风险管理，风险管理不只是项目经理估计风险，我们风险管理采用了全员的头脑风暴法，并按权重进行TOP10列表管理。针对TOP10风险，制定相应风险应对计划。在软件开发中，主要的风险有技术风险，所以风险应对计划是项目计划前期的技术验证和测试，需求不确定等风险的应对计划是原型和迭代的开发方法。例如在界面越来越重要的今天，需求规格中会做出原型界面并提前得到客户方的review是很好的风险应对计划 。

沟通是监督、控制的基础，是推动项目执行的基础，更是减少冲突的良方。有项调查项目经理90%时间在沟通。的确沟通占用了项目经理的大部分时间，因为项目经理是面对项目干系人最多的角色。在项目沟通方面，作为项目经理应周期性向机构管理层和客户报告项目的技术、进度、费用、质量方面的状况；在客户面前全面代表所在机构，与客户建立和维持友好和开放的关系，直接面向客户的项目经理是客户与所在机构最关键的联系点；做一个项目沟通的推动者、避免项目中出现沟通的遏制者；为项目沟通积极创造环境，包括集中工作；保证所有会议的高效率。项目经理要根据不同人员和不同情况下的问题选择最合适的沟通方式，来达到好的沟通效果。如果项目中出现与干系人等的问题，首先要检查沟通计划了。现在好多沟通事项是只装在项目经理的脑子中的，如何做好项目的沟通计划，是项目经理要培养成习惯的一个重要技能。这在我们软件项目中尤甚。例如员工流动始终是软件行业的一个显著特征，所以项目经理在处理此类问题时沟通计划就非常重要了。曾经我们一个项目开发中，一个team leader想离职技术移民到加拿大，由于前期不知是否能办好也不知道什么时候能办好，所以没有声张，等办好后，离离职就只有很短的时间了，而项目到了非常重要的时期，他们team成立不久，其他成员均无太多经验，他的角色暂时无法找到合适的替代者，我做了以下准备，一、通过朋友推荐和参加一些技术交流的机会招聘合适人员；二、通过老总，全公司内部挖潜，并建立推荐人奖励制度；三、由于平时大家关系很好以私人饯行名义和他沟通。告诉他现在他在项目无人替代的重要性，是否他有推荐的合适的人选替代，因为移民过去要保证半年时间在移民地，而找工作可能会有困难，而且他也表示并不想离职，代他和公司人事沟通，以重新修改劳动合同而不离职方式解决了他的后顾之忧，该teamleader也主动做好安排，延期出国，保障了项目的顺利结项和验收。我想，在奖惩、激励和重要沟通中选择合适的沟通方式会产生迥然不同的结局，尤其是我们好多软件开发人员比较内秀而不善言辞，项目经理更应注意这点，较好的引导和倾听能力是每个项目经理需要具备素质。

使项目成员目标一致，项目成员更好的进行配合和协调，有良好的沟通气氛、适当的竞争，减少冲突，提高项目组的工作能力和效率，团队建设是项目经理需要关注的一个课题，我的一些经验是尽早开始项目团队建设并持续进行；在项目计划、风险等重大决策上鼓励大家参与并取得大家的认同；经常评估团队绩效和效率；最好不要超过每周50小时的工作量；“利用好”老员工和上级领导。

经验分享。一项对项目经理的调查显示，项目经理平均每周参加6个会议，其中25%的时间浪费在无用的讨论上。会议效率低最普遍的3个原因是：会议没有很好的计划、会议没有被适当的领导、无纪律的与会者。我们软件项目也会遇到相同的问题，项目启动会、评估会、大大小小的评审会、技术会、周例会等等一系列会议会随着项目进展而召开，如何保证高效的会议效果，我的一些会议技巧与大家共享：确实需要开会时才开会；订立会议纪律；明确会议目标；提前准备会议议程；提倡各会议参与人的会前准备；鼓励参与，但在会议过程中遵守会议议程；把团队建设融入会议、作会议记录、会后跟踪所有安排任务的执行情况。

项目经理对项目成败负责，是项目中承受压力最大的一个，在风险较高的软件开发项目尤甚，项目经理沟通层面和项目可视化的工作繁复冗杂，计划中一定将管理活动列入工作量统计中，以便能使自己有时间和精力处理项目更重要的事情。项目经理应能忠实的反映一些无法力所能及的事情，取得上司的支持，这对项目成败非常重要。

软件项目管理，需要我们不但关注项目管理技术等在软件行业中的应用，还应该关注如何与软件新思想和技术的整合，例如XP等思想，使我们得到更高效益的产出。欲想琢其玉，必先利其器，项目管理和我们软件开发、质量管理等得一系列工具和模版，是我们事半功倍的利器。他山之石可以攻玉，关注一些管理界的发展，例如目前的中国式管理等，将其经验用于软件项目管理实践并总结，将为我们带来更大实效。

作者简介：
王晓毅，目前就职于益逊通（Esoon）软件公司任高级顾问，PMP，有多年软件项目管理经验。

■ 栏目编辑：熊妍妍（xyy@csdn.net）

来自死亡项目的验尸报告

■ 文 / 陈绍英

和产品开发一样，国内软件公司在做项目时，测试管理方面也存在很多问题。本文列出三个比较典型的现象，同时后面列举案例进行分析，最后讨论通常的解决方法。这几个突出的问题如下：

- 为了进度可以忽略一切，对质量的忽视比开发产品时还要严重；
- 短期利益行为严重，欺骗和愚弄客户。只要能通过验收“拿到钱”，不惜在客户面前回避和掩盖缺陷；
- 项目管理过程非常不规范，测试工作没有进行正确定位。

演示失败谁负责？

下面是一位测试工程师在某大型企业进行现场验收测试的经历，测试的对象是一个被拖了一年多工期的电子政务办公系统，项目验收时的基本信息如表1。

项目信息	内容
相关主要人员	测试人员1人，项目实施人员2人，销售经理1人（现场主要责任人），项目经理进行远程指挥。
功能模块	功能模块分有工作流和无工作流的两种，其中含有工作流流程的模块6个，包含了近20个业务流程；不含工作流流程的普通功能模块20个左右。
测试允许的时间	由于进度要求，测试的时间只有三个工作日
产品现状	含有流程的模块很多业务流程无法走通，非流程模块问题同样很多，产品基本不可以使用，用户无法试运行。

表1

工作过程记录如表2。

在第三天晚上10点多，开发和测试人员共同测试明天要演示的关键流程，在最后一个流程通过测试后，所有人员离开了用户现场。

在第四天的演示过程中，关键的一个流程在演示时，系统报出了一个“惊人”的错误，在文档流转过程中，单击“保存”，系统报错，流程无法继续正常流转，最终演示以失败告终。

尾声：在接下来的一周里，测试人员继续对产品进行测试，共计发现了150多个缺陷。再次进行演示时，基本取得了成功，当然演示时绕过了那150多个缺陷。

这类项目或者产品，应该被称为“死亡”项目。其基本特点是项目的工期被拖得无限长、人力等各种资源是预计的3倍

日程	测试人员	实施人员	销售人员	其他问题
第一天	(1) 测试了3个含有业务流程的模块，不含流程的模块6个，共计9个模块； (2) 测试结果为：大多数流程无法流转； (3) 提交测试报告，分为内部和外部测试报告两种。	(1) 上午自己检查模块； (2) 下午开始修改测试出来的缺陷； (3) 晚上修改完成了部分缺陷。	(1) 晚上把内部测试报告交给项目实施人员； (2) 外部报告发给公司和客户相关领导；	(1)实施人员由于另有任务，曾经多次在公司的Windows平台上修改缺陷，然后进行移植；而现场的环境是RS6000服务器的AIX系统，这也是导致缺陷较多的一个重要原因 (2) 外部测试是“报喜不报忧”的报告； (3)由于离用户现场较远，且随后需要进行演示，同时后到的实施工程师还要负责其它项目，所以第三天晚上大家力争早早结束任务。
第二天	(1) 测试了另外3个含有业务流程的模块； (2) 测试结果为：发现3个模块的7个流程无法流转； (3) 提交内部测试报告和外部测试两种； (4) 同时请求公司支援。	(1) 修改新发现的缺陷和前一天的缺陷； (2) 下午一位工程师被调走。	(1) 晚上把内部测试报告交给项目实施人员； (2) 外部报告发给公司和客户相关领导；	
第三天	(1) 全面测试含有流程的模块，到晚上仍然有6个左右的关键流程无法流转； (2) 前来支援的工程师测试不含有流程的模块；	(1) 一位实施工程师负责修改在不含有流程模块中发现的缺陷； (2) 晚上8:30分另外一位工程师重新赶到现场，修改6个关键流程的缺陷。	(1) 晚上把内部测试报告交给项目实施人员； (2) 外部报告发给公司和客户相关领导；	

表2

以上，项目中几乎每一个成员都成为了众矢之的。这类项目即使当时侥幸通过演示，在试运行阶段也会带来新的麻烦。缺陷在使用时肯定会被发现，只是时间问题。

成功当然无须赘述，但这类项目一旦失败就会成为关注的焦点。导致这类项目不成功的原因有很多，如进度管理、人力资源配备、项目的协调等各个方面都会把项目搞的像个“泥潭”。由于本书的定位，下面从测试角度对这个案例进行分析。在测试方面，主要存在如下问题：

测试报告：测试报告不应该只提交给销售经理，从而导致内部测试报告只有现场的实施人员能够看到。测试工作在行政上是第三方性质的，不单单要向客户和开发人员反应情况，更应该向自己的部门领导甚至技术总监报告。因此案例中的测试工程师首先要确定销售经理是否能把真实的测试情况反映到公司，否则应该按时报告发现的问题。

测试过程：在测试执行过程的最后，和项目实施人员共同进行测试，这是测试经常范的错误。测试不应该受到其他人员的干涉，如果按照开发人员的思路进行，大多数的功能能够通过测试。

测试资源：人力不足也是一个原因，尤其在进度目标确定时。提供保证进度的测试人力资源是个基本要求，否则根本无法完成任务。

在项目中投入的测试资源不足，已经成为一种普遍现象，尤其是在进度和成本压力大的情况下。这里推荐一个对项目进行测试的方法——产品线责任制，这种方法的中心思想是根据产品线设定测试项目经理，然后根据项目为其配备测试员进行测试实施工作。具体运行模式如下：

按照公司的产品分类来设置测试项目经理，每个主要产品对应一个项目经理。如果产品规模小，一个测试项目经理可以负责多个产品。同时这个项目经理对自己负责的这个产品要绝对的精通，包括产品的各个方面，他要有多次负责产品项目测试经验。测试项目经理初期的培养方式也很容易，可以让他长期负责一个产品的测试，在此基础上慢慢进行培养。

有了测试项目经理后，就可以给他配备一些测试员，形成一个测试小团队。当然，测试员是需要经常交换的，否则会产生测试中的“同化”现象（“同化”现象将在第14章中有详细的说明）。这里的测试员不需要有特殊的技能，招聘普通的测试员即可，主要是为了降低成本。

在具体的运行中，测试项目经理主要完成下面的工作：

1. 建立和维护所负责产品的缺陷跟踪库。这个缺陷库特别重要，是这个项目经理工作经验的总结，主要记录该产品在实施中常见的缺陷。测试员实施时可以针对这些缺陷重点进行测试，这样可以提高效率。测试项目经理也可以用文档列出测试重点，对测试员进行培训。

2. 对项目进行监督。每天要审核各个测试员提交的缺陷，必要时可以深入项目一段时间，和测试员共同进行测试。这样可以对项目的整体质量有大概的把握。

3. 培训。对执行项目的测试人员进行培训，提高其技能。

4. 协调。协调各个项目的人力资源。

5. 预测试。预测试核心功能模块，如果关键模块问题较多，可以暂停测试计划，由开发人员执行测试，以降低成本。

测试员主要工作是跟踪项目，同时进行测试。由于测试员的成本较低，完全可以让其从头到尾进行项目的跟踪。如果项目有空闲时间，测试员也可以投入到其它项目中进行测试。

这种方法还有一个好处是测试员也会在工作中越来越精通业务，逐渐提高技能，可以替代测试项目经理，减少了组织对某一个人的过度依赖。

通过初验的不合格产品！

本案例是一个典型的不成功的项目实施案例，不但耽误了工期，而且发生了项目实施过程中更换项目经理、项目组成员调动频繁等现象。尤其戏剧性的是项目通过初验即将试运行时，忽然发现质量不合格的。系统初验后，即将对客户进行培训的前一周，公司为了保证培训顺利进行，同时为了在培训后试运行时能有更好的运行效果，特意安排了一次测试，测试后发现仅在产品10多个核心功能模块中，就发现了近100个缺陷，而且其中很多都是低级的缺陷。

项目背景信息如表3。

工作过程记录如表4。

第三天晚上，得到公司批准的测试工程师坐上返回的列车。公司同时指示项目要进一步进行开发调试，开发人员完成自己的测试后，将会再安排测试工程师到现场进行验收测试。

这个项目的场景相信是国内很多中小公司做项目的真实写照。项目人力资源不足、拖延工期、几乎不投入测试力量等特点基本都得到了体现，甚至还出现了愚弄客户的情形——如此多缺陷的产

项目信息	内容
基本信息描述	本项目的任务是完成国内某省政府电子政务系统的实施工作，客户现场在外地。该项目在原有产品的基础上进行开发。 没有经过测试直接进行实施，并且成功的通过了客户的初验。
项目内容	共包含26模块，其中10多个核心业务模块。
进度信息描述	即将进入试运行阶段。
测试情况	在初验前，没有进行任何测试。为了保证培训和试运行的顺利进行，安排一名测试工程师在现场进行测试。

表3

日程	测试人员	实施人员
第一天	前一天夜里出发，在当天10点赶到现场，工作到夜里12点离开。共测试了10个模块，发现了50多个缺陷，其中的3个核心模块就发现了30多个缺陷。	当晚进行修改。
第二天	测试完成剩余的16个模块，发现了70多个缺陷。加上前一天的缺陷，共发现120多个缺陷。其中10个核心业务功能模块就有100多个缺陷。	继续修改新发现的缺陷。同时部分实施人员被到安排其它项目去工作。总共修改完成了20多个缺陷。
第三天	测试工程师以缺陷太多，达不到测试标准为理由申请返回，得到公司批准。	项目经理请示公司向客户申请延期培训，调派更多的人手修改缺陷。

表4

品通过了初验，无疑是动了一些“脑筋”。

测试的重要性不言而喻，由客户发现缺陷导致成本大大提高也成为共识，客户对产品进行投诉降低公司声誉大家也是心知肚明，可为什么大多数项目还是铤而走险的不好好测试呢？归根结底是没有良好的项目管理体系，没有进行正常的项目规划。测试计划应该直接纳入整个项目计划，测试工作应该从一开始就介入项目，因为在测试上节约的成本，一旦项目投入使用很快就会以数倍的投入再次被“花费”掉。

管理项目测试是一项更有挑战的工作，下面结合本案例分析一下提高项目质量的方法。

1、提高项目管理水平。按照项目管理规范来实施项目，这样就会正确处理成本、质量、进度三者之间的关系。项目的内容这里不做过多讨论，读者可以参考项目管理方面的书籍，这里只从测试管理角度来讨论分析案例。

2、树立“客户是上帝的观点”。尽管“用户是上帝”的口号提出了几十年，但是在软件企业把这句口号落实到实处的屈指可数。这诚然与中国软件行业的大环境有关系，但“以买方为导向的软件市场”的形成也是一种必然趋势。案例中如此多问题的系统能够通过初验，相信在初验的演示中一定是回避了那些问题。毕竟计算机是诚实的，只要有缺陷，最终还是会暴露出来的。因此，“老老实实”的解决掉所有问题才是上策。

3、规范测试流程。项目由于进度压力，往往忽略测试，尤其在后期。实际项目上线后遗漏的缺陷将会逐一暴露出来，这个时候的成本投入往往更大，涉及技术支持、缺陷验证、现场修改等一系列的工作投入。

4、加强版本管理。项目投入使用前的版本还好些，但是由于赶进度等因素也会发生问题。产品投入使用后，是版本最容易混乱的阶段。实施工程师经常在现场解决问题，如果没有进行版本维护，就会造成公司内部版本和客户现场版本不一致的情况。这种版本混乱往往越来越严重，甚至版本失控。因此，版本控制也是提高项目产品质量的必要手段。

5、严格的缺陷管理。这个项目在最后撤回了测试人员，无疑缺陷管理又失控了。缺陷管理和版本管理是相辅相成的，只有都控制好，才可以做好项目的质量控制工作。项目开发出的产品，本身控制缺陷就很有难度，因此需要加大力度。具体控制方法，可以参照第八章的缺陷控制流程进行管理。

什么才是好的测试策略？

项目的基本信息如下页表5。

尽管这个项目完成了任务，但是从执行过程中暴露的问题可以看出，测试策略制定、贯彻和执行等仍然存在许多问题。在讨论测试策略之前，我们先了解一些项目管理的有关知识。《项目管理知识体系指南》(PMBOK)中采用如下文字描述项目：

项目是为完成某一独特的产品或服务所作的一次性努力，具有下面三个特点：

● 一次性：

项目有明确的开始时间和项目结束时间，而且：

■ 不一定历时短；

■ 项目提供的产品或服务通常不是一次性的；

■ 市场机会稍纵即逝，项目需要在限定的时间范围内提供产品或服务；

■ 项目团队具有临时性。

● 独特性

项目要完成的工作是以前没有做过的，项目要交付的产品或服务与其他的产品或服务在某些方面有显著的不同，所以是独特的。

● 渐进明细

“渐进”意味着“一种分步骤，持续稳定增加的过程”，“明细”意味着“工作需要仔细、详细并通盘考虑”。产品或服务的特征在项目早期被粗略和广泛的被定义，随着项目团队对产品的进一步理解和认识，这些特征会更为精确和详细。

项目开发工作是一种特殊的产品开发工作，要针对具体的用户，进度压力非常大。测试工作是整个项目工作的一部分内容，如果把项目看成一个大项目，那测试工作就是一个子项目。因此，项目测试工作可以按照项目管理基本思想来进行。(PMBOK中大项目定义：通常由若干个有联系的或类似的项目组成，以相互协调的方式对其进行管理，以获得单独管理每个项目所不能取得的预期收益。有时也指规模特别大、时间特别长的项目。)

项目管理过程包括项目启动、计划、执行、控制、收尾五大过程组，涉及到范围管理、成本管理、进度管理、风险管理、人力资源管理、沟通管理、采购管理、质量管理以及整体管理九大知识领域。如

项目信息	内容
项目描述	1.是一个第二期项目（在一期中实现了用户需求的部分功能；二期中实现另一部分的相关功能） 2.项目人员变动比较大；测试负责人为参与一期、比较熟悉需求的成员 3.由于一期中用户和开发人员对测试评价很高，所以二期中，在测试资源需求上以及对测试的重视程度上都得到了很大的外部支持。
项目相关主要人员	测试人员 3 人，另有其他项目人员 8 人 系统使用角色比较多（50 多个）；
功能模块	主要业务流程为 3 个；辅助业务流程为 3 个 功能模块 60 多个
测试允许的时间	4 个月
产品现状	一期得到了用户的认可；开发组比较重视测试
测试结果及经验	
总体来说，本项目中在平稳中进行，最终很好地完成了测试计划。整个项目在下面各个方面执行的比较好： ● 测试负责人全程参与了需求 ● 详细地编写了测试案例 ● 测试负责人严格跟踪了测试进度 ● 有效地选择了测试策略 ● 严格地进行了测试版本控制	
测试执行过程中的一些问题	
缺少明确的测试目标，或者说是对测试目标执行的力度不够。使测试组没有去考虑更先进的测试技术和方案，而一味地使用旧有的管理模型，导致了测试案例设计阶段资源宽松的现象，造成了测试内容不够全面而且有严重的人力资源浪费。而到了测试执行阶段，出现了时间短、任务集中的现象，导致测试时间非常紧张，甚至于无奈采取放弃部分测试执行的策略，无法达到比较完善的测试。 从项目整个的资源投入来说，本身的人员安排已经达到了 2.5:1 的比例，而且人员投入比较早，所以在资源安排和时间安排上是比较充足的。	

表 5

果把整个测试过程看成小项目，也应该按照项目管理过程来管理。因此，做好测试策略工作的重点就是平衡质量、进度、成本的关系，然后再进行其它方面的工作。本书针对目前国内大多数公司没有足够的人力资源来完成测试工作的现状，提出下面在项目测试时制定测试策略的一些指导思想：

质量：对项目的质量没有必要过于苛刻，只要能满足用户使用就可以。例如一些校验、边界值、界面方面的测试，如果进度压力大，只要可以使用、用户能够接受即可，没有必要把软件“千锤百炼”后再交给客户。当然，如果有时间，应该进行更充分的测试，可以概括为一句话：“千方百计、最大限度的提高产品质量”。质量永远是重要的，因为如果软件的质量用户不接受，其它工作可以认为没有意义。

进度：进度压力大是项目的突出特点，因此要力争满足进度要求，甚至可以牺牲一部分质量或者加大成本投入来保证进度。

成本：赚钱是软件项目的目的，因此压缩成本是必须的，而且要尽可能最大限度的来压缩成本，压缩成本同时还要考虑质量和进度。赚钱是做项目的目的，除非了为了加强客户关系，或者是为了得到新的项目这些长远的利益，否则花费的成本一定要控制在盈利之内。因此，测试经理对成本的把握就是成本投入要满足测试项目质量要求。

测试策略的制定是一项难度较高的工作，空谈一些理论还是无法制定出正确的测试策略。这种能力需要经验的积累，测试管理人员要具备下面的知识：

1.业务能力：做好测试策略首先要了解测试对象，可以通过不断的学习和积累达到。测试管理人员要在具体执行工作中提高业务能力，否则自己都不知道要测试的是什么，还谈什么制定测试策略。

2.工作经验：工作经验在评估成本的时候非常重要，合理的计划是测试工作正常开展的保证。

3.项目管理技能：测试项目本身就是一项项目管理工作，因此需要掌握相关的项目管理技能才可以做好项目测试工作。

具备了上面的能力后还要充分考虑指定策略的一些注意点：

1.根据项目的客户需求、项目方案等来制定测试目标，从而制定出正确平衡质量、成本、进度三者关系的策略方案。

2.对项目的风险要做详细的分析，还要考虑项目客户情况、资源情况等其它背景信息。

3.与项目经理多多沟通。项目的发展变化是动态的，通过项目经理掌握的动态项目信息，就可以不断调整测试策略，使测试策略能够和项目变化相适应。

4.获取更多客户方面的信息。例如客户比较苛刻、计算机基础很好等信息无疑提出了更高的质量要求。进度压力大时，可以考虑在升级或者使用过程中来不断提高质量。

5.评审并公布测试策略。策略是一种指导思想，首先评审需要得到公认，同时还要灌输给所有相关测试人员，这样才能保证实施的顺利进行。

总之，制定测试策略一定要根据实际情况，“一切从实际出发”永远是对的。

■ 责任编辑：欧阳璟（ouyangjing@csdn.net）

网站项目成功管理实践(上)

■ 文 / 刘振飞

【编者按：本文作者刘振飞曾在《程序员》今年第1、2、3期上连续发表了访谈文章《Bug管理的经验和实践（上、中、下）》。不仅Bug管理，刘振飞认为微软公司软件研发成熟的产品（项目）管理经验可以推而广之，应用在IT产品研发的各个领域。他离开微软后，先后就职于国内两家中小IT公司，直接主持过两个网站项目的研发，取得了成功的实践。本文将他今年的网站项目http://133.newsky.cn作为一个学习的典范，以飨读者。】

在开始做http://133.newsky.cn之前，我已经明白网站的开发与产品开发没有什么不同。不过在2004年离开微软中国研发中心Office组的时候，我对网站开发仍一无所知，这主要是因为我之前没有任何互联网研发的背景。虽然对传统软件产品的研发管理比较有经验，但从未接触过Internet相关的项目。

从零开始与网站开发亲密接触

去年我接手第一个网站项目http://www.okooo.com开发时，并没有做网站的经验，只能试着按照以前我参与做Microsoft Office时的方法来做：

首先是打造一个便于公司内部沟通交流的内部网，其中包含“传统软件”研发需要的三个工具：文档库（存放公司各项目的文档）、CVS（保存项目的各种源代码）、BugFree（记录项目的各种缺陷）；

然后，抓住“需求、开发、测试”三个环节：

● 要做好规划、明确需求。为什么要做这个网站、要达到什么目标？特别是需求，要详细到每个页面的每个区域放置什么内容。网站需求应该由对业务最熟悉的人来定义，他负责按照我要求的规范（详细程度）来写出每一部分需求文档，并放入文档库中。每完成一个页面定义，我就召集开发、测试人员来阅读、讨论，这样全部需求写完的时候，项目组成员对整个网站就有了一个清晰的认识。

● 需求明确才进入开发阶段。首先是定义数据库——有多少张表、每张表中有多少个字段。我和开发组长反复讨论，搞清楚这些表定义能否涵盖全部需求，这是最关键的一步，决定着下面编码能否顺利进行。数据库定义后，就是网站后台管理的编码实现，也就是对一张张表进行管理（增、删、改）。当后台管理完成时，项目的大部分就大功告成了。用户看到的前台页面仅仅是内容展示——把一张张表中的数据取出来按照最初的需求放置到页面的各个位置。所有的代码都用CVS管理起来。

● 网站测试和开发同步进行。后台管理每完成若干张表的管理，测试人员立即开始测试。这就像流水线，开发完一部分，立刻测试；同样的，网站前台展示开发时也一样需要测试人员跟进。发现的每一个Bug都用BugFree记录下来跟踪处理过程。

● 数据统计跟上。网站后台各个表的任何改动要准确记录，决不允许出现不知道谁修改了数据库内容的情况。其次，网友访问网站的日志要做好统计，每天结束的时候就能准确的看到当天的用户访问数据。这些数据对网站运营极其重要。

四个月后，我的第一个网站项目顺利上线。所有参与该项目的同事感觉都很新鲜，因为以前他们在做网站时，基本上是一个人“包干”一个频道，简单构思一下就开始写程序、边写边想、相互独立。后来，我跟一位曾在某门户网站工作过的高级工程师朋友介绍了上面的做法，他非常认同和赞赏，得到他的认可我也很兴奋。

随后接触到的很多网站技术人员，让我发觉作坊式做法同样存在于互联网公司，网站在重复多年前传统软件的老路：一个“大虾”很厉害，搞定一个频道或一个网站的方方面面，离开他谁都玩不转；代码中处处留着他的灵感，人走了，网站维护就成了大难题：没有文档、没有统一的编码规范、没有测试记录。

其实无论传统软件、网站、还是游戏等等软件产品/项目，都是程序员用一行行代码敲出来的，只要像微软软件研发那样抓住需求、开发、测试这三个环节，其管理都极其类似。因此当我进入http://133.newsky.cn网站项目的时候，信心十足：我能把它管好！

打造一个网站开发的品牌项目

05年2月18日：项目启动，开始整体规划

在我加入金环天朗的时候，这个网站就已经存在了，而最开始的计划也只是对原有的网站进行局部改版。但是等我深入了解后，大吃一惊：

◆ 规划／需求：原有网站没有经过认真规划就匆忙上马，只有部分的简单示意图，对于每个页面具体区域的功能描述和逻辑过程还是依赖口头沟通。没有独立的后台管理，依赖于WAP业务的后台，内容展示力不从心。

◆ 页面设计：美工因为还有其它工作所以有一定程度的拖延，没有时间观念，整个设计方案没有经过整体评估，导致后来许多细节没有按照计划实现，页面设计先后由两人分头独立完成，导致部分风格不一致。

◆ 开发：技术实现一直处在救火的状态，没有规划，没有步骤，没有主次之分，没有时间观念。代码的结构非常散乱，没有可用的文档查询，开发人员走了，给以后接手的人带来极大的麻烦。代码没有规范、没有注释。归结起来就是可读性很差。

◆ 测试：没有任何测试，开发人员简单试一试就直接上线了！

◆ 内容：网站内容维护没有专人负责，逐渐处于无人管理的状态。

总之，原来的网站有太多不尽人意之处，和同类网站比起来差距较大，市场人员无法推广，技术人员很难维护，动不动就出错。只能另起炉灶，推倒重做一个全新的网站。

对一家SP公司而言，做网站是打通让用户消费的通道。从常远看，内容为王，但短期内通道为王：就是让用户很容易找到公司提供的内容。因为WAP业务非常依赖于运营商的门户排名，一个业务放在运营商WAP门户上，第一屏和第二屏有着本质的不同，愿意翻到第2屏上的用户可能少一半或更多！所以SP要想尽一切办法来摆脱对门户的唯一依赖，必须能用别的通道让用户很方便的找到你的业务。而网站就是最好的宣传通道，是公司产品最重要的展示平台。网站研发的目标就是尽快打通联通、移动用户的消费通道，把公司生产出来的产品（图、铃、文字）方便地展示给更多的手机用户。

这个http://133.newsky.cn网站是面向中国联通用户的，其设计目标是：

▲ 1～3年内不需要改动大框架

▲ 公司业务内容的精美展示、销售平台

▲ 在同行中有很强的竞争力

▲ 老板可以拿出来给投资人演示

为了达成这个设计目标，我和项目组花了近一个月的时间来制定完整规划。

明确的研发流程应该是一个开发团队的固定资产，从这点上，我建立了一套项目研发流程，并为其提供工具支撑：

▲ 认识老网站的现状、确定新网站

	规划	需求	美工	开发	测试	运营
2005-2-18	收到老板Email，项目启动					
2005-3-22	完成规划	启动前台展示需求的定义				
2005-4-04		开始后台管理需求定义				
2005-4-12		完成需求定义。确定后面的时间进度：6/15正式上线运营！	开始后台管理页面设计	开始网站数据库的设计		
2005-4-15				完成“后台管理详细设计”的文档		
2005-4-16				开始后台管理的编码		
2005-4-21			开始前台展示页面设计			
2005-4-26				完成后台管理的编码		
2005-4-28					引入测试组，开始后台管理的测试	
2005-5-10				两名新人到岗，开始前台展示的编码		
2005-5-23						确定运营组成员及分工
2005-5-31				主要编码结束		
2005-6-08					测试完毕	开始录入内容
2005-6-12						内容全部上线
2005-6-15	http://133.newsky.cn正式上线运营，向公司全体同事通报！					
2005-6-22	完成Postmortem(项目总结)，为下个移动网站项目做准备					

丰富的资讯、翔实的内容）；明确人员分工、并预告项目执行的几个关键点。

▲ 在没有公司内部网的情况下，我先搭建两个工具：用于保存各种文档和源代码的TortoiseCVS(客户端)+CVS(服务器端)，用于缺陷管理的BugFree。为每个项目搭建一个CVS模块，其中都有四个子目录：Spec(需求文档)、Design(设计文档)、Code(源代码)、Test(测试文档)。



示意图：网站项目的 CVS 目录

然后是人力资源，我在规划中提出了非常明确的人力资源需求：

▲ 前台需求定义：1 人（蔡志宏）

▲ 后台需求定义：2 人（刘振飞、朱伟波）

▲ 美工设计制作：1 人

▲ 开发：3 人

▲ 测试：5 人

▲ 运营：5 人

然而，当时的情况却是项目组人员迟迟无法到位：美工只有一个兼职的、时间无法保证；只有一个开发组长；没有测试人员；网站运营人员不能确定。针对这样的情况，我的任务还包括了招聘相关人员及时到位。

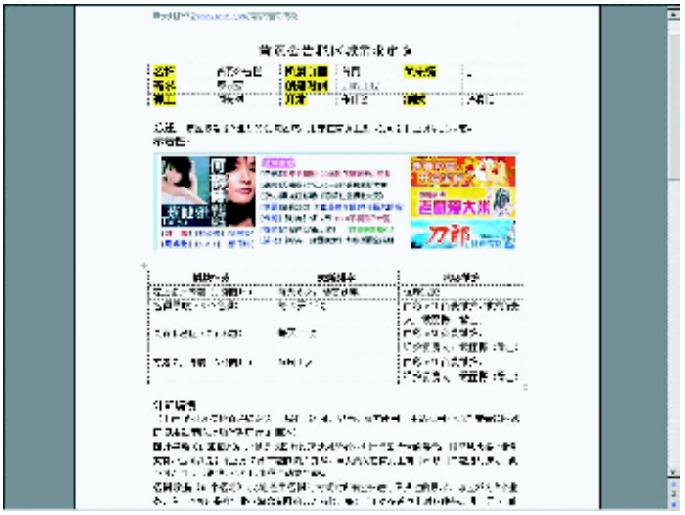
在整体上完成上述工作以后，时间已经是 3 月 22 日了，事实上，在整个项目启动初期的准备阶段是一个非常重要的工作，清晰的项目规划也为后来的工作扫清了很多障碍。这就是俗话说的“磨刀不误砍柴工”。

05 年 3 月 22 日：开始定义网站需求

网站需求特别难以确定，为了解决这个问题，我将整个需求定义划分为三个主要的部分：

1. 网站前台展示的定义

我首先和负责定义需求的蔡志宏确定了需求 Spec 文档模板，然后他根据首页、二级、三级页面逐个页面、逐个模块的去定义：展示什么内容，大概的模样（最终样式由美工负责）。这样每个页面都被分解成一块块的“部件”，一个“部件”由一份 Spec 描述，比如下面是“首页公告栏区域需求定义”Spec的示意图。



示意图：首页公告栏区域的需求定义 Spec

每完成若干相关联的 Spec，我就召集美工、开发人员开会讨论（本应该也叫上测试和运营人员，但当时还没有人），大家站在不同的角度去看看有无问题，并最终确认下来。

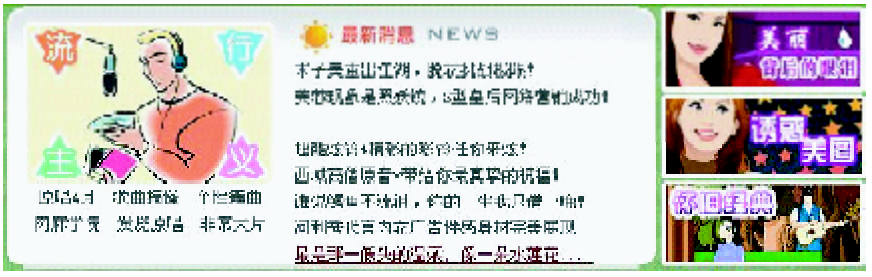
2. 联通用户消费流程的定义

用户消费流程涉及到收费问题，必须把每个细节都要搞清楚。这个需求由我负责，先形成一份PPT文档，在大范围内征求大家的意见，然后细化每个细节：从用户访问我们的首页开始，如何登录，如何转向联通网站，如何扣费等每个细节必须想到。

3. 网站后台管理的定义

根据网站前台的需求，我和开发组长朱伟波来设计数据库定义，确定多少张表、每张表中有什么字段。然后从运营人员的登录页面开始设计，用PPT把每张页面的示意图以及逻辑关系都展示出来，然后把需求、开发、美工召集起来一起讨论，看看是否符合运营人员的习惯、是否有遗漏的地方。

需求文档要想清楚后再写下来，让别人读得懂。定义好的需求 Spec 是整个项目开发的“合同”，马虎不得。在需求定义的3周中（其中前台展示的需求用了2周、后台管理的需求用了1周），每写出来若干相关的需求文档，就在项目组内讨论一次，最终明确下来。需求文档一旦成型以后，就必须严格按照需求文档编写设计代码，尽量控制需求的变化。这不但要求我们在最开始的需求分析阶段做好最充分的准备工作，而且还需要作为项目经理的我，顶住一些来自各方意见的压力。幸运的，我们团队还是非常好的坚持下来了：



示意图：上线后的首页公告栏区域——完全根据 Spec中的需求定义来实现

然而，另外的一个问题是，需求文档很容易“老旧”、跟不上最新的变化，需求定义人也懒得去更新，因为开始编码后谁都不去注意需求文档了。为了解决这个问题，我就在后台管理的每一张表的维护页面上，增加一个“Spec”按钮，点击后就可以看到相关的需求文档Spec列表了。这样做有两个好处：一方面运营人员可以很方便的看到最初是怎么设计这块功能的；另一方面也把需求定义者的工作暴露在全体同事面前，文档写的好坏是一目了然。

充分的需求定义保证了整个项目能够准时完工，这也是我们这个项目能够取得圆满成功的原因之一。需求确定之后，后面开发、测试的时间就基本明确下来了。

05年4月12日：数据库设计，正式编码开发

有了完整的需求文档后，接下来就进入开发阶段。如同前面提及的，首先需要完成的是数据库的设计。其实早在需求定义期间，我和朱伟波就已经开始数据库定义，确定多少张表、每张表中有什么字段。我们花费了三天左右的时间来对后台数据库进行详细的设计，并产生出设计文档。

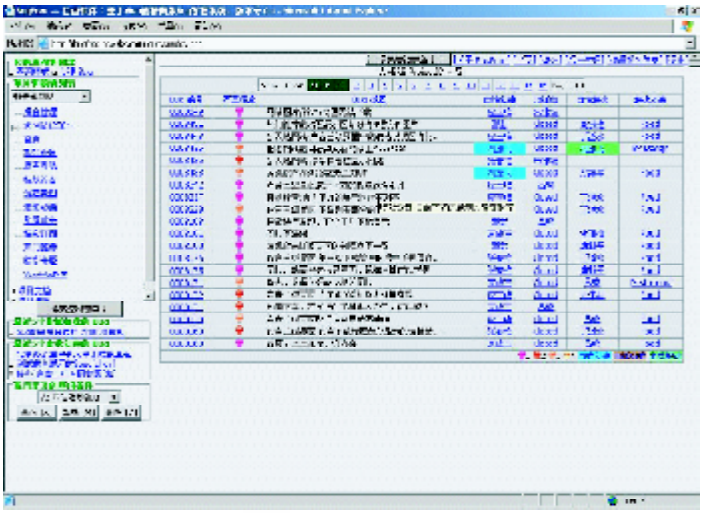


示意图：新天地网站2005版后台数据库定义.doc

然而，光有需求和详细设计文档还不够，开发团队需要保持要一种一致的风格，这一点要求所有的程序员对代码有责任感。因此在这个阶段之前（3/16~4/12），我就让公司所有的Java工程师多次讨论，并最后确定一份“编码规范”，这样网站真正开始写代码的时候，就有一个明确的规范来约束代码的书写。

对于软件项目来说，经常会有一些出乎意料的情况发生。比如，本来计划有两个开发人员做后台管理，结果因为沈阳联通的一个合作项目需求紧急配合，只好临时抽调一个人去支援，毕竟网站是公司内部可以控制的，导致后台开发只有朱伟波一个“光杆司令”，那一段他连续十余天加班到晚上11:30！这样高强度、高压力的工作状态，不是每个程序员都能承受的。经过朱伟波的努力，终于在十天时间内将所有的后台编码全部完成（4月16日至4月26日）。

紧接下来，从5月10日开始，专门为网站开发配备的两个Java程序员到位，朱伟波首先给他们介绍后台管理和Struts技术。随后分工：首页、3个铃声二级、3个图片二级、杂志、热门推荐、精彩专题等，每人承担几个频道的实现，分头阅读对应的需求文档并随时找蔡志宏讨论不清楚的地方。当理解需求之后，由朱伟波协调，三个人分头开始进行前台展示页面的编码工作，加班加点，在5月31日基本结束。

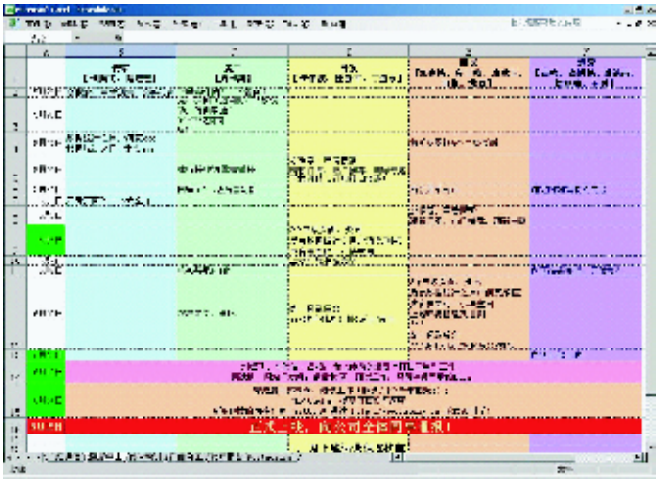


示意图：用BugFree来管理网站项目中的Bug

05年4月28日：与开发并行的测试

在这个项目之前，整个公司是没有测试人员的！这不得不让我大为惊讶，一个SP公司没有测试怎么行！所以在这个项目进行的同时我启动测试人员招聘工作，最终成立了一支5人组成的测试组，负责所有业务的测试。

当网站后台管理编码完成后，4/28立即启动测试工作：后台管理中的首页管理、动画、声音、彩图、专题、资讯由专人负责测试，发现一个问题就在BugFree中记录一个Bug。通过BugFree的跟踪和记录，可以让某些问题不必累积到最后才解决。随着网站前台展示开发在5月中旬启动，测试工作也在并行跟进：每个频道、每个页面都有专人负责检查，这样尽可能的把各种潜在的问题揪出来，免除后患。



示意图：项目最后突击的日志

在此期间，整个项目组都进入了最后的冲刺阶段。为了确保6月15日网站能够上线，我开始将工作日往后倒推，每一周甚至每一天，需求、美工、开发、测试、运营等环节需要到达什么状态，必须做到心中有数；每天都要盯着进度，一旦有了延误，必须立即找到原因和补救方法。如果实在忙不过来，我就要做出决定砍掉哪些可以延缓的功能模块。

很遗憾的是，因为测试组搭建的比较晚、测试任务又比较重，他们需要花费很长时间去熟悉公司的各种业务，所以在这个网站项目中，对测试文档部分（比如测试用例）我并没有要求，只要把问题发现出来上Bug就好了。这就是项目管理中的Trade - Off：抓住主要矛盾、抓大放小。这个项目结束后，测试组已经逐步成熟、磨合好了，我才开始强调测试文档的重要性，每个业务测试时一定要同步完成相关的测试文档（计划、用例、测试结果等），测试时就按照相关的测试文档进行。这样以后复测就能省掉很多时间，换个人测试也很方便上手。

经过一个多月的努力，测试组的同事基本上完成了网站所有频道、页面的检查工作（6月8日）。

05年5月23日：迟到的运营组开始运转

研发人员做出来的网站只是一个空空的框架，没有实际的内容填上去，网站就无法上线——打个比方，研发人员把“大楼”盖好了，还需要运营人员把“内部装修”做好。然而面对人员的稀缺和内部调整，一直到5月23日才确定运营组长及组员分工，他们匆匆进入角色，一面熟悉网站后台管理，一面准备内容。6月8日才开始正式的网站内容录入。

05年6月15日：网站正式上线！

值得庆祝的日子到来了。6月15日凌晨我正式向全公司同事报告这个网站正式上线。这是我自己主持研发的第二个网站，也是我非常用心管理的一个项目。我想留下一个参考样板，为公司其他项目的管理摸索经验。我认为这是一个成功的项目是因为：

◆ 做出来的网站符合最初的规划和需求定义；

◆ 按照需求定义完成的时候（4月12日）确定的进度向前推进，6月15日上线是两个月前就确定的；

◆ 整个项目执行过程中，规划、需求、开发、测试等环节均按照预定轨道前进，没有出现大的纰漏。

整个项目组成员在网站上线后都非常兴奋，这应该是公司到目前最成功的一个项目管理实践。公司领导对这个项目的研发表示非常满意。现在的情况是，休整2周后，7月4日按计划我们又启动了第2步移动网站的研发；另外对历史遗留的众多WAP业务的整理开始提上议事日程。我需要更多时间、耐心和细心，和需求、开发、测试等各个环节的同事们密切配合，把公司各个业务的项目都理出头绪来、让研发部为公司业务的不断成长做出贡献，也让技术人员工作“累身不累心”，不要总是“救火”，要能看到辛苦工作的成效。

网站开发和产品研发没有什么不同！

按我整理的时间表和项目计划，对照微软的流程，你会发现，我完全是按照微软“传统软件”的研发流程去管理这个网站项目，略有不同的地方是，这个网站项目的时间跨度比较小（只有4个月），而且人力资源有限，美工、开发、测试三个环节我只能是并行处理、流水作业，以尽量缩短项目的整体时间。

我也算是“把微软先进的软件研发理念和中国中小企业的具体情况相结合”吧，其中最难的是把项目研发流程的理念灌输给全组同事以统一认识，并能有效的执行下去。很多时候要靠我不断的去PUSH各个环节，做的比较累，但在完成之后，很有成就感，尤其是针对一个团队不断发展和成熟，所做的努力是显而易见的。（未完待续）

■ 责任编辑：欧阳璟（ouyangjing@csdn.net）

	规划和需求阶段	开发阶段	测试阶段	发布阶段
主要与人	Planner 与PM驱动	开发人员推动	测试人员推动	PM，产品经理，运营管理层执行
阶段成果	目标描述（Vision） 详细需求文档（Spec） 日程进度表	M1, M2, ... Code Complete	集成测试 Bug-Fix, Check-in Dogfood Beta1, beta2, ...（Triage） Zero Bug Release	Show-Stoper bug Release Candidate(RC) Sign-off RTM（Ready To Release）

调查：方兴未艾的管理咨询

—— CMM/CMMI 在中国

■ 文 / 中国系统分析员顾问团 徐锋 张友生



面对方兴未艾的 CMM / CMMI评估认证热潮，常常让人想起 ISO 在中国的历程，“ 形式主义 ”是否仍然在重现？通过本刊调查，您将重新认识管理咨询热潮。

软件过程成熟度体系 CMM / CMMI 伴随着‘ 印度模式成功神话 ’传入中国之后，深深震撼了中国软件行业。几年来从尝试走向狂热，又从狂热走向了平静，CMM / CMMI在中国绘制了一条什么样的曲线？有多少企业寄身于这个热潮？通过评估的企业获得了什么？CMM / CMMI为中国企业带来了什么？现在对它的认识是什么样的呢？哪些咨询机构被大家所认知、认可呢？

带着这样的问题，希赛®网软件工程频道（http://51cmm.csai.cn）举办了一次从用户的角度，对面向 IT 企业 CMM / CMMI咨询和评估市场的咨询机构进行大范围的“ 中国CMM / CMMI咨询机构10强调查 ”。本次调查是以完全的用户视角描绘 CMM / CMMI 咨询和评估市场的群英谱，展现行业生存态势，聚焦企业的真实需求，梳理行业变幻格局。

本次调查历时 80 天，本次调查历时 80 天，数据库中共有45178条调查记录，管理员按照下列原则对数据进行了筛选和优化：

■ 凡是重复投票的，则只保留一条投票记录。

■ 投票人职位、部门、地址完全一样，且记录大于等于3条的投票，则只保留一条投票记录；

■ 投票人 IP 地址相同的，则进行人工审查，逐条核查信息的真实性，剔除恶意拉票成分。

按照以上原则处理后，剩余41093条有效调查记录。

大势研判

从对这次调研的结果数据分析可以发现，近年来中国软件业对于软件过程管理以及改进的意识和需求已经逐渐成形，这也给CMM / CMMI咨询机构提供了施展自身的广阔市场机会，但要走的路的还很长，要做的事情还很多。

● 软件企业已经认识到软件过程的重要性与改进的必要性

中国的软件业起步较晚，而且加上国内市场环境成熟度相对较低，因此‘ 作坊式 ’还是企业的主要生产方式，绝大数企业人员规模小、市场面窄、生存环境较差，个人英雄主义还在盛行。而印度模式的成功神话给中国软件业当头一棒：优秀的人才比我们少、内需市场比我们小，却迅速地实现了跨越。这一个以‘ 规范的软件过程 ’实现‘ 化腐朽为神奇 ’的案例，强有力的例证了“ 软件过程 ”的威力。

● CMM / CMMI 咨询机构尚未走出“ 一头热 ”的现状

虽然企业认识到了，但是却因为自身规模小、投资能力差，承接的项目规模小、心有余力不足，团队成员水平不一、风险大等种种因素，还无法马上迎接 CMM / CMMI。因此，近几年来进行CMM / CMMI认证评估的企业还都限于规模较大的那部分，而且它们的目标也各有不同：

◆ 拿政府的钱，为自己贴金：虽然在 CMM / CMMI评估热的前期，各地政府均出台了不同的优惠和鼓励政策，因此各地的规模较大的软件企业都纷纷响应这个“ 投资少又能赚名誉 ”的好事。但这时的许多企业领导者还有许多对 CMM / CMMI到底是什么，能有什么用都还没有

搞清楚。

◆ 为自己的业务拿牌照：印度神话传入国内后，激增了一大批从事软件外包的软件企业，有的是转型、有的是新创建，而CMM/CMMI这个“从业执照”也就自然而然成为香饽饽。

◆ 少数企业是真实地为了软件过程的提升：虽然也有一些软件企业的确是为了对自己的软件过程进行改进，但仍然属于少数。

面对这样的市场，咨询机构当然只能够是“一头热”的现状，而且也必然难以避免为了迁就客户而出现一些折衷。

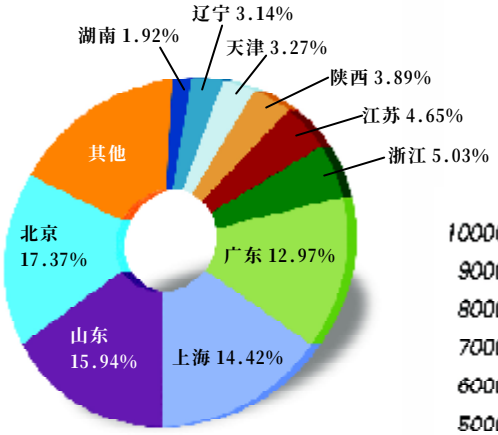


图 1:被调查者区域分布表

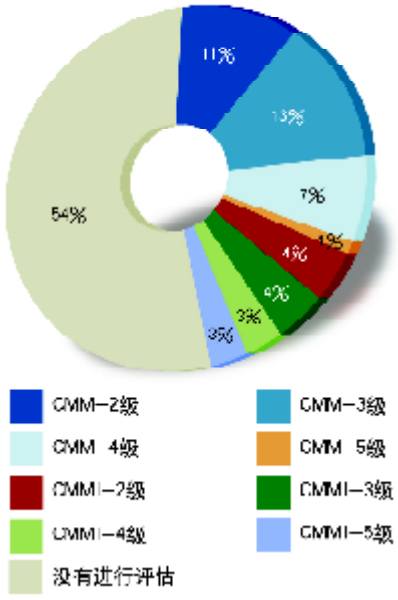


图2:贵公司通过了CMM/CMMI 哪个级别的评估？

● 务实的软件过程改进需求将是下一步的市场机会

但现在CMM/CMMI狂热的阶段已经过去，逐渐进入了平静期。人们有了更多的认识与思考，也从第一批通过评估认证的企业中看到了正面的、负面的榜样。

毕竟，软件过程改进是一个正确的方向，是产业提升的必由之路，因此软件企业仍然存在着这方面的需求。不过，这时的需求会更加趋于理性、更加趋于务实。因此，如何面对这样的市场机会，值得咨询机构多一分思考。

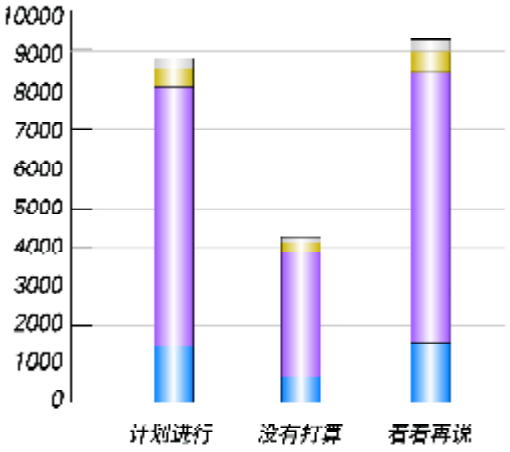


图 3:如果贵公司还没有进行 CMM/CMMI 评估，是否有打算进行该项工作？

数据分析

接下来，我们再从本次调查得出的数据结果中挖掘背后的信息与故事，更深入地了解这个市场的实际情况。

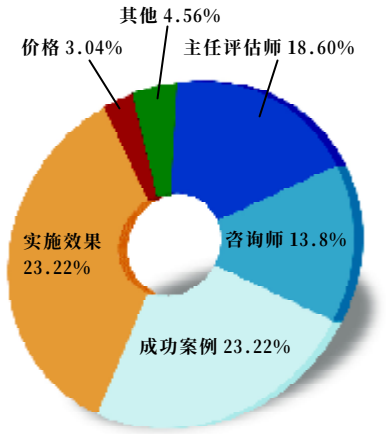


图 4：贵公司选择 CMM/CMMI 咨询和评估机构首要考虑的因素是什么？

◆ CMM/CMMI 关注者与IT的产业布局直接相关

从图1中，我们可以发现参与调者的对象主要集中在北京、上海、广东、山东等软件业相对发展较好的地方。但这一带的CMM/CMMI服务也比较普及了，那么如何挖掘其它地区的潜力的，是一个值得思考的问题。

◆ 市场潜力仍然巨大，但挑战不小

从这一组图，我们可以得知：参与调查者所在的公司还有54%没有进行认证评估（图2），而且这部分的潜在市场中80%都有计划（一半在计划，一半在观望）（图3），这无一证明CMM/

排名	咨询机构	分析
1	员工有抵触情绪	1）新的工作方式对传统流程改变大 2）企业中技术导向性强 3）技术人员大都关心技术，不关心过程
2	持怀疑态度	1）形式主义的“前辈”造成了不良影响 2）CMM/CMMI如何更好与中国现状结合有待研究
3	领导支持力不够	1）领导者对软件过程的认识不够 2）软件过程改进的实效体现有待研究

表 1 推行 CMM/CMMI 体系的主要困难分析

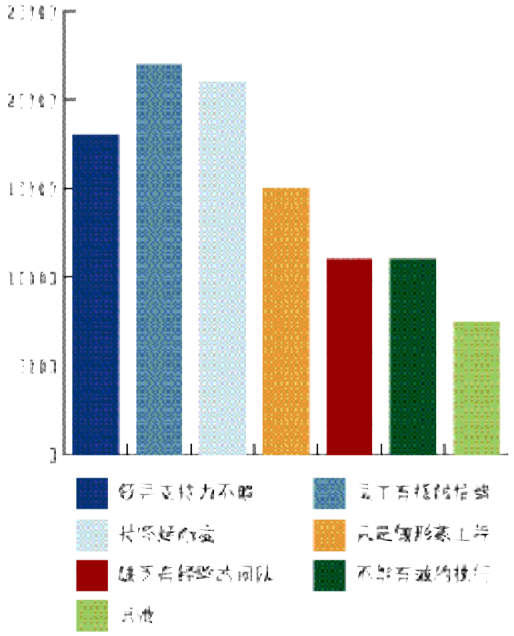


图 5：您认为推行 CMM/CMMI 体系的困难有哪些？

CMMI 认证评估的市场潜力仍然巨大。

而从图4中我们可以发现，现在的软件企业对于咨询机构的选择，关心的重点内容是：实现效果、成功案例、主任评估师和咨询师等涉及咨询机构自身实力的因素上了。这也充分体现了“务实之风”已经开始，也对咨询机构提出了新的要求，“没有金刚钻，不揽瓷器活”。

◆ 软件过程改进的阻力源于内部

从图5中，我们可以发现阻碍CMM/CMMI体系的困难还是主要源于软件企业内部。而且从中也不难发现，以前的一些“形象工程”给其造成的影响不小，而且产业的稚嫩也是需要面对和解决的问题。

◆ 咨询机构百花争鸣，各有特色

在本次调查CMM/CMMI的10家咨询机构中，大多都有自己的长项，受到不同用户的认可。下表中体现了本次调查各机构的得分情况。

结言

面对方兴未艾的CMM/CMMI评估认证热潮，不免让人回想起ISO 9000在中国的历程和现状，“形式主义”的跟进，是否还有几分“大跃进”的身影，但这种做法必然只能给中国的企业带来更大的创伤。CMM/CMMI虽然不是万能的，但毕竟给予了中国软件企业走向规范化、规模化、成熟化创造了一个良好的契机。

中国软件企业能否有效地把握，踏实地改进自己的过程，提高自己的竞争力，任重道远。作为这一提升的催化剂，CMM/CMMI咨询机构承担着沉重的历史使命。只有通过具有使命感和责任感的咨询机构精心的帮助，具有务实精神的软件企业共同努力，才能避免步入ISO的后尘，才能够真正促进中国软件业的全面提升。

声明：本调查报告的实质是作者对参与调查者问卷的总结和归纳，其中的观点不代表希赛®网。

作者简介：
徐锋，《程序员》专栏作家，系统分析员，CSAI 顾问。自 1992 年开始就被电脑神奇的魅力深深吸引，一发不可收拾。从 Apple II的汇编语言开始入门的我，一直就在计算机软件领域孜孜不倦、兢兢业业，不敢有一分懈怠。十余年来，涉猎面广泛，只叹自己仍博而不精。现致力于需求工程、系统分析与设计、过程改进等领域的研究与探索。

■ 责任编辑：欧阳璟（ouyangjing@csdn.net）

表 2 咨询机构评价表

序号	咨询机构	主任评估师	咨询师	实施效果	价格	服务	总分
1	北京斯福泰克科技发展有限公司	3.98	4.33	3.75	4.05	4.88	20.99
2	北京赛柏科技有限责任公司	4.53	4.10	4.27	3.72	4.24	20.86
3	循序咨询（上海）有限公司	3.75	3.57	3.92	4.07	4.01	19.32
4	赛宝认证中心	3.65	3.45	3.64	3.50	4.38	18.62
5	北京新世纪认证有限公司	3.00	3.32	3.44	3.82	4.37	17.95
6	北京翰通融达管理顾问有限公司	3.73	3.41	3.70	3.35	3.75	17.94
7	北京英普信科技公司	3.73	3.36	3.62	3.39	3.43	17.53
8	北京中信保科技发展有限公司	3.60	2.83	2.96	2.37	3.56	15.32
9	北京赛迪国软认证有限公司	2.84	2.97	3.01	3.95	3.49	15.26
10	北京瑞贝泰格管理咨询有限责任公司	2.98	2.64	2.87	3.14	3.61	15.24

表 2



Microsoft为何不能按时交付产品？

2005年6月1日——Longhorn、SQL Server、Visual Studio都延期了

文 / Jennifer deJong

Microsoft软件推迟发布已成家常便饭。但是没有人真正回答过该公司为何不能保证承诺的交付日期。

分析师和Microsoft社区成员透漏了一些内情，大部分得到了该公司的确认。

“集成，集成，还是集成。”纽约市Jupiter Research的资深分析师Joe Wilcox说。对于Visual Studio 2005，数据库、开发工具、操作系统和企业服务器比以前更紧密地结合在一起，从而造成了交叉相关性。“如果SQL Server推迟，Team System也将推迟，”研究公司Gartner的集成和开发副总裁Tom Murphy在谈到Visual Studio 2005生命周期开发版时补充

说。“如果设计的系统互相依赖，有好处，但是也让这些系统难以改变。”

每日构建（Daily Build）的困难

创建这类系统的过程带来了问题。Visual Studio 2005（以及它所依赖的产品）已经达到了一个极限点，“每日构建和测试整个系统对于Micfosoft来说已成为一个挑战，”Murphy说。“他们已经进入了这样一个阶段，[曾经]能够管理发布的标准[开发]实践不再奏效了。”他还提到了Jim McCarthy的著作“Dynamics of Software Development”，该书提倡按时交付技术，如每日构建。

被问及标准开发实践是否不再有效时，Vsiual Studio和开发人员平台产品管理的负责人John Montgomery简单地说，“构建具有战略重要意义的软件非常复杂。仅仅Visual Studio 2005，我们已经交付了从Express产品系列到Team System等14个产品。另外，我们正在致力于把公共语言运行库结合到SQL Server中，这本身就是一个相当可观的工程。”

Team System将延期真是莫大的讽刺，Patrick Hynds指出。他是一位Microsoft地区主管和CriticalSites的首席技术官，这家.NET咨询公司位于马萨诸塞州的伯灵顿。Team System就是为了加强沟通以便按时交付软件项目而设计的，他说。“设计，生成代码，修改代码，然后再返回去按照变更要求编码。”Microsoft在三月份宣布将在“2005年下半年”交付Team System（和Visual Studio 2005的其他版本）。而此前该公司曾承诺将在“夏季”推出。Microsoft的一位女发言人后来又宣称三月份的声明并不表示会延期。

说出真相

既然Visual Studio 2005如此浩大，为何Microsoft不计划好更加现实的交付日期，或者等到产品准备好之后再宣布呢？该公司没有回答这个问题，但是有人提出了答案。

“他们为什么不调整产品发布规划呢，我不知道，”Jonathan Cogley说，这位Microsoft MVP for C#是Thycotic Software的CEO，这家专门从事敏捷软件开发的咨询公司位于华盛顿特区。（MVP，或者说微软最有价值专家，由Microsoft为表彰其参与Microsoft社区的活动而授予。）“也许是Microsoft有意事先宣告吧，他们实际上并不在乎，”Dan Appleman推测道，他是位于San Jose的Desaware的总裁，该公司为Microsoft开发人员提供组件。“不过我并不在意是否延期，只要做得好，做得可靠、做得安全就行了。”

安全正好是Microsoft的首要目标。“他们不希望一个新产品因为不够安全而遭到遗弃，”CriticalSites的Hynds说。“对于每个人来说安全都至关重要，”他补充道，“是否要让产品比市场上的其他产品更安全，即使对于伟大的设计也是一个困难的抉择。”

Microsoft的Montgomery说，安全绝对是整个公司最关注的问题。但他没有说是否因为解决这个问题而造成了延期。

Microsoft可能还遇到了法律问题。今年年初的一份报告指出，公司官员会见了司法部的监督员，他们要求Longhorn保证遵守2001年通过的美国反托拉斯法案。该报告推测，这些会晤可能是Longhorn延期的部分原因。

当有人问到满足DOJ或者欧洲相关规定的要求在多大程度上推迟了Longhorn和其他产品的交付时，Montgomery说：“在Longhorn开发过程初期，我们一直密切、坦诚地与DOJ合作以解决各种怀疑和问题。”

即使反托拉斯法没有直接影响到延期，也可能改变了Microsoft的行为。“Microsoft小心地避免被人怀疑[自己]出轨了，”Hynds说。他推测可能是因为法规而不是兴趣制止了Microsoft购并其他公司。“银行里存着600亿美元，他们完全可以买下Macromedia这样的公司。”但是这样做将面临危险，他说。Adobe已在四月份宣布收购Macromedia，这一无所不在的Flash播放器的开发商。

潮水般的反馈意见

过去谈到软件延期时，Microsoft常常解释说“要对客户的需求负责”。这种答案毫无新意。不过社区成员指出，和过去相比，Microsoft在请求更多的意见反馈，从早期的beta版、社区技术预览（CTP）到bug追踪竞赛这类活动等等。

“反馈的意见非常之多，就像是从消防水龙头上喝水一样，”来自Thycotic的Cogley说。“过去18个月中，MSDN的所有订户都曾经看到Whidbey被砍掉了四五次，”Whidbey是Visual Studio 2005的代号。如果客户不喜欢他们看到的东西，Microsoft就“返回去重做”，Hynds补充说，“这是一把双刃剑，反馈越多意味着又要返回到设计中去，从而推迟了交付日期。”

Microsoft的Montgomery说，潮水般的反馈意见对交付时间产生了一些影响，“但我认为这是完全值得的。”

这些理由仍然无法解释为何IBM以及其他公司能够按时交付软件开发工具，而Microsoft却不能的问题。IBM按照预定的交付日期，于2004年12月推出了它的生命周期开发产品Rational Software Development Platform。“这样比较是不公平的，”Gartner的Murphy说。IBM从Rational那里得到了平台中包含的很多工具，而Microsoft则从头创建用于Team System的建模、测试和其他工具。

Desaware的Appleman说，虽然感觉上好像Microsoft推迟得比过去更久了，实际上并非如此。“我用过的每种Microsoft技术都曾经被延期，”他说，“我记得向COM过渡的时候，谈到延期，我们认为VB4永远也不会出来了。”

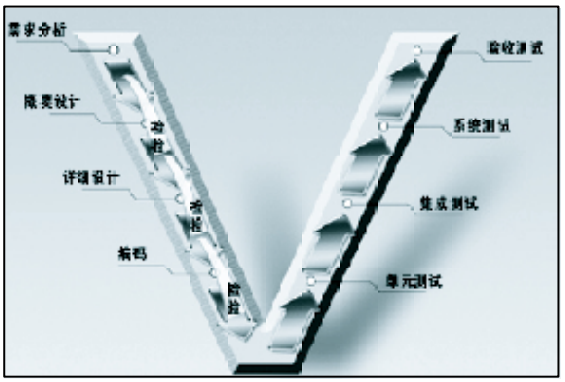
■ 责任编辑：欧阳璟（ouyangjing@csdn.net）

测试过程的监控方法

■ 文 / 贺炘

软件开发项目的成败，很大程度上取决于三方面的配合：过程、人、技术，三方面相互制约，又相互促进。为了能更加有效地管理软件开发项目，规划软件开发过程，近年来国内引入了不少软件开发模型，如：CMM/CMMI，RUP，XP等，每一种都体现了一种思想，都希望能在最大限度内，协调上述三者之间的关系，最大程度的减少软件开发过程中的风险，能按照既定计划，交付出合格的软件产品。

对于软件测试工作，国内大多数企业采用V模型作为测试的标准模型，来规划和设计软件测试流程，指导日常的测试工作，模型如图一所示：



图一：软件测试V模型

通过这个模型，我们发现按照理想情况，如果需求获取人员完成了需求分析工作，测试人员就可以按照需求分析的结果规划我们的系统测试工作，设计系统测试用例，等待到系统测试阶段执行测试用例，验证系统是否实现了设计的所有功能和性能要求。

通过分析我们发现，按照V模型来设计测试工作，测试人员可以在前期（需求获取阶段）就介入到整个开发过程中，设计测试用例规划测试工作。这样，有许多工作就可以并行开展，而且很多问题可以在开发的前期被发现，极大的规避了开发工作的风险，降低了改正缺陷的成本。

但是我们目前的实际情况是什么？需求总在变更，概要设计做的不够好，而且变化频繁，详细设计不够详细或者根本不做，单元测试覆盖率不够或者根本不做，这样造成测试工作步履维艰，质量难以控制。我们上面谈到的几种软件过程改进模型，也是想在方法的高度上，尽量的改变这种现状。

不管我们打算采用何种模型作为我们过程改进的基础，对于软件测试工作来讲，V模型都是我们很好的一盏路灯，它提供给我们一个软件测试工作提前介入的思路，以测试或者说以质量保证为前提的软件开发方法，只有这样做，我们才有可能生产出高质量的软件产品。

本文并不打算一一探讨上述几种过程改进模型的测试监控方法，而是参考V模型的架构，从软件项目管理的角度谈一谈，如何对软件测试工作进行监控，具体的监控手段都有那些，在平时的工作过程中我们应该怎样使用。

大家都知道，项目管理有三个要素，即：成本，进度，质量。

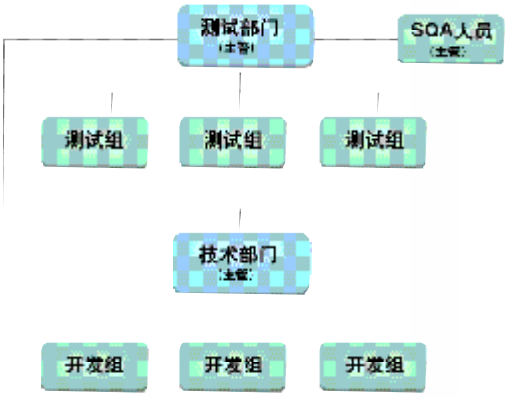
对于软件测试经理来讲，只需要对产品的质量负责。对于整个项目来讲，项目经理作为项目组的最高领导自然要对项目整体的：成本、进度、质量负责；在这个团队中，作为主管软件测试工作的测试经理，需要协助项目经理只对质量负责，这样才能客观的对项目的质量做出评价。之所以说不用对其它两项负责，更确切的说法应该是在做质量判断的时候，不需要考虑成本和进度可能对质量造成的影响，具体的权衡工作由项目经理或者公司的高层来完成，测试经理只提供对软件产品质量的客观判断。

既然测试经理只对质量负责，这就会衍生出来一个问题，测试经理对产品质量过于吹毛求疵，与开发人员造成对立，进而影响项目开发工作。如果这件事情发生了，有一个确切的信号已经传递了出来：测试人员和开发人员在沟通上存在问题。如何解决这个问题？首先，我们应该审视测试人员和开发人员的沟通技巧是否存在问题。其次，我们应该重新核对我们在项目开始时确定的质量目标，看看是测试人员人为拔高质量目标，提

出超范围的要求，还是开发人员人为降低质量目标，生产出不符合质量要求的产品，以此作为对质量标准实施误差的一个判断。

为了能有效的降低这种风险，管理上采用的一般方式就是监控，即由第三方人员对被监控方的工作进行客观的评价。那谁是第三方人员？首先，这个人不在被监控的项目中负责具体的工作，其次，他代表公司或者所在部门，需要对项目的质量情况进行客观的评价。

针对一个具体的组织结构模型，如下图二所示：



图二：组织结构图

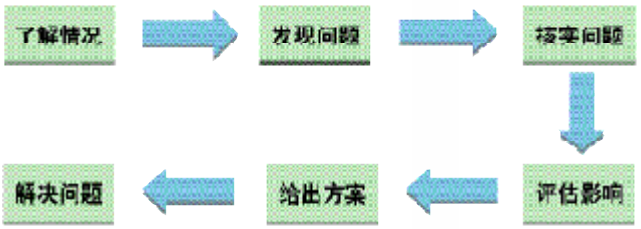
可能对测试工作有监控需求的部门有：软件测试部门的主管，SQA人员或者其主管，技术或者开发部门的主管。他们的监控出于不同的目的，如：评估测试工作的有效性，了解具体项目的质量情况，了解开发的进度和效率等。不管出于什么目的，他们有一个共同的特征是：不参与项目组中具体的工作，并且需要在短时间内了解项目的实际情况，并且做出相对准确的判断。但是，不在项目组中，对项目组的实际情况不是非常了解，如何能在短时间内对项目的测试情况做出准确的判断？

在实际工作过程中，我们可以采用如下的方法对测试工作进行监控：

一般采用的手段是：问讯与查阅相结合，对关键点进行抽样审核，并询问不同的人员以进行核实。

具体的审查点和查阅项会在下面做详细的阐述。在监控过程中，我们一般会经历如下(图三）阶段：

1. 首先依据自己的经验，问讯项目组的相关人员，看看项目的过程是否符合通常的测试规范。



图三：测试监理过程

2. 通过问讯，记录发现的问题或者疑问。

3. 通过询问不同的项目组成员或查阅相关的文档，核实发现问题或疑问的真实性。

4. 汇总所有问题，评估各个问题的影响和风险，列出优先级。

5. 给出可能的解决方案。注意，这里的解决方案不是指具体的解决方法，而是指激发项目组成员行动的可行的方案，如建议项目组开会讨论可能的处理方式等。

6. 跟踪解决方案，验证问题是否真正得到解决。

以上是一个通行的监控过程，这里需要强调的一点是：不管出于什么理由对测试过程进行监控，但是发现问题绝对不是我们的目标，能够有效的解决问题、降低项目的风险才是我们的目标，

对测试工作监控的方法，依据项目所处的不同阶段我们分为三个部分进行阐述。

这三个部分暂且称为：测试初始期，测试实施期，测试结项期

测试初始期

在这个阶段，测试工作刚刚启动或者才开始按照计划实施测试工作，测试工作的启动时间点在各个公司可能不同，有可能是：需求调研后期，集成测试期或者系统测试期等。具体在软件开发的什么阶段测试工作开始介入，并没有一个一定的说法，关键要看所在公司的软件开发活动的成熟度来灵活选择，但是这点并不影响下面的讨论。

作为测试工作的监控者，应该在那些重要环节上加以注意？首先请大家注意这个阶段的特征：软件开发工作已经开始，需求开发工作已经完成或者接近尾声，以测试人员为主的测试工作正式启动或者刚开始运行。

在以上的前提下，我们来说一说需要注意的监控点：

● 测试工作有没有明确的工作范围

这是在测试工作中最需要明确的，也是非常多的项目忽略的工作。在做测试工作之前，一定要非常明确准备对那些内容进行测试，预计达到的质量标准是什么，尤其是对那些不进行测试。

我遇到过的很多的测试经理都抱怨说：“我们不可能在前期把这些事情都弄清楚，开发人员都不知道产品将来是什么样子，我们怎么知道需要测试那些内容？”咋一听，感觉很有道理，但是情况是否真像大家说的那样？作为公司，或者项目经理都希望能将项目做好，能生产出一件令用户满意的产品，如果这个假设成立的话，这也就是我们能够改变现状的动力。

首先，原先的那种做法已经多次证明是行不通的，所以只有改变我们的做法才有可能成功，前期先弄明白我们打算做什么并不是一个过分的要求。

其次，开发人员实际上并不是完全不知道他们打算生产什么样的产品，而

是就一些细节考虑的不够，或者不周全。作为测试人员，一定要知道如何对一件产品的功能和性能怎么验证，这实际上在帮助开发人员从使用者的角度上重新的审视一遍需求，也许这时候开发人员也说不清楚，那如果你和开发人员都非常清楚哪些是明确的、哪些是需要后面再补充的，也已经达到了我们的目的。

● 被测系统有无明确的性能指标？

对性能要求比较高的系统，需要在前期明确具体指标到底是多少？用何种手段进行确认？用户是否认可这个指标的描述以及确认的方法。

性能指标一般从需求中对一些敏感的数字的描述中来，如：必须保证能处理30个在线用户同时操作，主要业务系统响应时间不能大于1.5秒等。

针对这些数据，测试人员一定要细化数据背后的含义，使这些数据变得可验证。如在规划这些性能指标的验证方式时，首先需要明确软硬件的环境是什么？在此基础上，还需知道什么叫30个在线用户同时操作？都操作什么？这个场景应该怎么模拟？只有和这个指标相关的所有验证方法都可行，而且得到了认可，在后续的测试活动中我们才能相信这条性能指标能够进行测试。

● 测试工作有无明确的阶段划分（如单元、集成、系统、验收等）？各阶段是否有明确的交付确认条件？

实际过程中，我们都会将测试工作按照阶段进行划分，上面描述的是一个通常的划分方式。在做监控工作时，还需要明确一点：这些阶段的划分是不是只有时间点的描述，而没有各个阶段之间可量化、可衡量的交付确认条件？

如果只有时间点的划分，那我已经可以断定，这个项目势必会延期，原因是在整个生产活动过程中没有明确的阶段点交付的检验标准，问题肯定会沿着整个开发过程逐步的传递下去，终归会在某一点爆发，最不幸的爆发点是在客户处。

如果想尽量的避免上述的风险，就需要在开发过程中明确各个阶段点之间的交付、确认条件。而且这些条件必须可量化、可衡量，决不能是含糊的，不易操作的，否则在实际操作过程中还是会将大量的问题推入下一个阶段。

下面也以单元测试为例，看看怎么建立一个可量化、可衡量的单元测试阶段的交付标准。

首先，需要确定开发人员是否进行了单元测试。可以让开发人员提交一份单元测试总结报告，上面需要大致的描述一下进行了哪些单元测试。单元测试总结报告是否提交是一个可以量化的条件。

其次，需要评估一下单元测试的质量，主要可以通过如下方法：

> 是否有足够的单元测试用例？可以对照详细设计规定单元测试用例的数量，这是个量化的方法。
>
> 单元测试用例的通过率必须达到90%以上。
>
> 测试人员还可以抽样执行开发人员编写的单元测试用例，抽样执行的单元测试用例的一次通过率必须在90%以上。这也是一个量化的方法，同时检查了测试用例书写的质量和单元测试执行的质量。

以上是一些常用的方式，而且这只是非常少的一部分，当给出确实可行的方法和可以量化的指标后，我们发现，评估一个产品是否达到了预计的质量要求就会变得相对容易很多，而且也避免了人为主观判断的尴尬。

● 最终的交付验收有无和客户确认通过的验收标准和范围？上线、割接和维护期有无明确的成功、失败判定标准？

对于项目类的软件开发，上面的监控非常重要。为客户做项目的时候，一定要在前期和客户确认怎么进行验收，验收通过的标准是什么，最好能形成书面的文档，这样才能在最后交货的时候避免不必要的麻烦。而且，验收标准和范围应该是测试的一个最小测试集，要最大程度的确保正确性。

如果是比较大的软件开发项目，还会牵扯到：上线、割接、维护，一般会写在前期的合同中，客户会按照上述阶段点阶段性付费，所以如果上述阶段点没有明确的成功、失败判定标准的话，对于公司尾款的收取是个挑战。

通过以上的详细询问，我相信，测试工作范围界定的是否有问题，测试工作是否规划的全面、细致，监控者应该比较清楚了，下面的任务就是将你的疑问记录下来，留待后面做证实。

测试实施期

在这个阶段，测试工作进行了一段时间，测试人员的工作应该已经步入正轨，按部就班的完成一些任务。这个阶段的特点是：开发人员和测试人员都按照日常的规范开始有条不紊的工作，有可能对一些问题已经习以为常，或者已经被同化。作为测试工作的监控者，应该在看似合理的工作中找出影响质量的问题，规避风险。

在这个阶段，应该关注以下问题。

● 缺陷管理流程是否规范？每个缺陷的提交和关闭是否都有复查？

缺陷管理是贯穿于整个软件开发过程、测试过程的关键环节，也是测试工作的根本，所以缺陷管理的流程是否规范，将是监控的重点。

首先，需要询问测试经理，软件开发过程中对缺陷的实际管理情况是如何的？不要让测试经理背诵公司的管理规范，而应该以一个实际缺陷为线索，追寻这个缺陷的产生直到关闭的过程是什么？期间是否有相关的记录，证明项目组的实施过程完全与描述相一致。标准的缺陷管理流程是怎样的，这里就不做

叙述了，如果大家有兴趣可以查阅相关的资料。

在这个过程中，还需要注意一点：缺陷的提交和关闭是否都进行了复查。

缺陷提交和关闭的复查人可以是测试经理，或者测试经理指定的人选，一方面经过复查，可以减少缺陷的重复提交，提高缺陷报告的质量，另一方面在测试组中会有一个人对系统或一个大组件的质量情况有比较全面的了解，尤其在后期，这种了解会在很大程度上降低系统误发布的风险。还有一个好处是，在测试人员和开发人员交互的过程中，这个复查人员起到了桥梁的作用，可以有效的隔离开发与测试之间的多头沟通，在一定程度上提高了效率。这个角色可以是专职的，也可以是兼职的，关键看系统的大小。

● 配置管理工作是否规范？测试过程中涉及到的版本是否都可以完整的追溯？测试版本的发放频度是否符合测试的实际要求？

配置管理工作是整个软件开发过程的生命线，相比较而言，开发人员对此应该更为关心。对于测试人员来讲一方面要保证可以取到自己想要的文档版本，另一方面必须得到自己关心的程序的任意一个测试版本，以便可以在正确的版本上执行正确的测试用例。

在实际检查过程中可以在缺陷库中任意选择一个缺陷，查看这个缺陷是在那一个版本的程序中发现的，随即在配置库中调出该版本，看是否可以调出。随后，查阅该缺陷在那一个版本中修订正确了，随即也在配置库中调出该版本，看是否可查到。

在这个过程中，还需要注意开发部门提交给测试部门版本的频繁度，看是否过快或者过慢。过快或者过慢，没有一个时间上的判断。比如每2天提供一个新版本供测试人员进行测试，这个是过快还是过慢？判断的依据关键要看测试人员所处的状态，当版本提交的过快时，测试人员一直忙于对已修订好的缺陷进行反测，没有时间对新功能进行测试。当版本提交过慢的时候，测试人员的时间比较空闲。

在监控过程中，只需要询问测试人员的测试工作的紧张程度，一般就能够判断出版本提交的频度是否有问题了。

● 关键测试活动的关键测试资源是否如期到位？如没有到位是否进行了合理的规划来完成延误的测试工作？

在测试过程中，某些关键测试任务需要用到特殊的设备或者特殊人员的技能，称为关键资源。在测试实施过程中，要提前计划会用到那些关键资源，以免耽误项目进度。

作为测试的监控者，需要非常关心这些关键资源的使用情况，因为如果关键资源不能如期到位，势必要影响项目的整体进度。

如果由于某种原因，关键资源没有如期到位时，要注意测试人员是否对计划进行了修订，修订的结果是否可以弥补已经造成的损失，或者能最大程度的减少损失。

● 测试策略，测试计划，测试方案，测试用例是否都经过了正式评审？发现的问题是否都进行了更正？

作为测试的监控者，不可能在短时间内评估一份测试计划制定的是否合理有效，一份测试方案是否可以正确实施，并且也不必要这么做。

测试策略，测试计划，测试方案，测试用例等文档都是测试过程中的关键文档，也直接决定了测试工作的质量。监控者在评价这些文档的质量时，首先想到的一点就是我要充分的阅读这些文档，以我的经验和能力来判断这份文档的好坏。但是，作为一个项目组以外的人，很难能就所有的细节发表高质量的看法，其次，也不可能在短时间内完成所有文档的评价工作。所以这不是我们的解决方案。

在监控过程中，首先要相信项目组自身的能力，假定他们有能力完成这些工作，这样工作就简单了，也变得可以操作了。

首先，查阅这些文档，大致看看，有没有明显的问题。其次，应该检查这些文档的评审记录，看看相关的人员是否参加了该评审，都发现了什么问题，大家的意见和建议都有那些。最后，看看所有的发现的问题是否都得到了解决，文档是否按照解决的方法进行了修订。

在监控的过程中，默认参与评审的人员技术能力都符合要求，这样只需要关注评审的过程就可以控制质量了。但是，如果有证据证明，评审的人员或者组成不符合要求，作为监控者应该宣布该文档的评审无效，需重新进行评审，以解决问题。但是，使用这项权利的时候要小心，而且要充分论证，否则会扰乱项目组的正常次序。

● 测试的相关文档是否都按照项目目前的实际情况进行了更新，并严格遵照执行？

经常会听到一句话就是：计划赶不上变化，这个问题就是冲着这句话来的。我在讲课的过程中问过很多人这样一个问题：不做计划，直接做事情行不行？至今我还没有遇到一个说行的。但是，如果计划和行动不同步，这个和没有计划又有什么区别？

项目中有各种各样的理由告诉你，我没有同步计划是合理的，但是我们的要求一定是必须有计划，而且必须严格遵照执行，这才是降低系统风险的唯一合理方法。

● 项目先前定义的测试范围在后续的计划、方案中是否有遗漏？

在测试初始期我们一直强调测试范围的必要性，在测试实施阶段还需要检查前期规划的测试范围是否在后续的计划活动中覆盖完全了，只有计划中完全的覆盖了所列的测试范围，才能保证系统的质量。

● 项目的测试过程是否按照公司预计的测试过程执行？

在测试实施阶段，还需要了解测试人员是否按照公司要求在执行所有的测试活动，但是要在短时间内了解，手段只能是听测试经理陈述他们的测试过程，再加以判断。如果公司有SQA人员，工作就相对简单了，只需要到配置管理库中找到SQA的检查报告，这些疑问就一目了然了。

测试结项期

在这个阶段，主要的测试工作已经进行完毕，最终的发布版本也已经准备出来。测试经理开始书写最终的测试报告，申请发货。

作为测试的监控者，这个阶段的主要任务就是评估软件产品的质量，依据已有的数据评估测试工作是否做到位，产品是否可以发布。

在这个阶段，应该如何进行监控，问些什么问题？

● 测试中发现的缺陷趋势曲线是否处于收敛状态？各个分模块的缺陷趋势曲线是否基本一致？

测试完成后，判断产品是否能够发货的一个重要条件就是：缺陷趋势曲线处于收敛状态，并且持续一段时间，表示系统处于稳定状态，满足发货条件。

那为什么还要看各个分模块的曲线是否一致？因为，有的系统比较庞大，有可能某一个局部的缺陷曲线还没有处于收敛状态，但是整个系统的缺陷趋势图已经把这个信息掩盖掉了。所以，还需要分别看一下各个模块的趋势曲线，确保系统的每一个部分都处于稳定状态，这样发货的风险才能降到最低。

● 是否有评判产品能否发货的文字性材料？

发货前，测试经理或者项目经理必须提交一份整个系统的整体质量说明，以文档的形式证明整个系统质量稳定，达到用户要求，可以发货。

在这个过程中，如果和客户有关于质量的约定，还需加入，如：用户签字认可的验收报告，用户签字认可的性能测试报告等。

● 是否召开了正式的最终评审会议？会议的参与评审人员是否有公司主管的高层？是否有用户或者能体现用户方意见的人员参与？所有的遗留问题是否都有了明确的解决方案，并且有相关的责任人负责问题的解决和跟踪？

在发货前，还需要召开正式的评审会议，而且会议必须有项目组以外，主管该项目的公司高层和能体现用户方意见的人员参加。因为，一般系统中或多或少都会遗留一些缺陷，这些缺陷到底应该如何处理，会给公司和客户带来多大的麻烦，都应该在这个阶段做一个评估，以决定该产品是否能够发货。

当一个问题确定遗留在系统中后，还需要对这个遗留问题有个明确的解决办法，如：在升级版本中修改，建议用户用以下方式绕过，或者干脆不再进行修改，都应该有一个明确的答复，而且还需要指派专门的人员跟踪问题的解决情况，并进行报告。

● 在测试初始期确定的结项条件是否都得到了满足？

为了能保证系统的正常交付，在前期和用户做了一些交付的约定条件，在这个阶段，需要确定这些条件是否都得到了满足，并有相关的证明。

如：性能指标是否满足用户要求，用户是否签字确认。验收测试是否完成，用户是否签字确认。后续的试运行期的方案是否完成，用户是否同意，是否有明确的截至条件等。

总结

以上以一个测试监控者的角度，探讨了如何对软件测试工作进行监控。希望本文对大家的日常工作有些许借鉴意义。

笔者在本文的最后还想强调几点：

● 监控是管理部门一项日常工作

这件工作要在平时不停的进行，才能有效的改善产品的质量，而不是一时心血来潮的意气之举。

● 监控的目的是为了解决问题，而不是发现问题。

监控的目的不是为了证明我们的能力，而是为了能够持续不断的提高软件开发的能力，能够持续改进，所以发现问题并不是目的，解决问题才是根本。

● 对问题的判断要准确，可验证

作为监控者，同时也代表公司对这件事情进行的判断，所以当发现问题的时一定要判断准确，而且经的起复查，这样才能避免不必要的麻烦，才能将更多的精力投入到处理事情本身的工作中。

● 质量的提高，工作的改进是一个渐进的过程，绝不能一蹴而就。

对于一个不太成熟的项目组，通过监控，可能发现很多的问题，这时不能太急躁，要心平气和，一点一点的改进我们的过程。

● 通过分析，要先解决重要的事情，要有总体的规划，而不是头痛医头，脚痛医脚。

发现的所有问题不可能都改正，这就要有个总体的策略，从大处招手，以公司的整体能力提升为要点，去区分解决问题的优先级，有节奏、有计划的将公司引入持续改进的轨迹，逐步的提升公司的核心竞争力。

作者简介：
贺炘，测试时代（TestAge.net）工作室成员。
北京慧灵科技有限公司高级顾问，测试培训讲师，ISO9000注册外审员，北京软件行业协会测试工作委员会副秘书长。先后在北大方正，神州数码等国内知名企业从事测试管理工作。

■ 责任编辑：欧阳璟（ouyangjing@csdn.net）

鱼与熊掌

—— 完美主义架构师的梦魇

■ 文 / 柯化成

完美主义等于瘫痪——丘吉尔

世界上有完美的建筑么——我相信有的，站在吉隆坡双塔（K.L.C.C）或是巴塞罗那圣家大教堂(Temple de la Sagrada Familia)前时，我找到了答案。



图 1：KLCC 照片

世界上有完美的软件么——我相信有的，使用Donald Knuth设计的“千金求一BUG而不可得”的TeX排版系统时，我找到了答案。

建筑大师贝聿铭说，“建筑家和艺术家不同，他受到许多方面的制约，他必须创造出一种生活的氛围，而这种氛围必须适应时代。因此他是为时代的发展而工作，而不是创造自己的小天地。”

那软件建筑家——架构设计师呢？

Trade-off 1：Time or Space

时间效率与空间效率是计算机科学的永恒命题——时间效率与空间效率。从冯·诺伊曼（Neumann，John von）体系被发明的那天起，关于时间和空间的‘折衷’就无处不在。也许，只有等到下一位图灵（Alan Turing）般的人物出来登高一呼，发明新计算模型，才能解开这个结。

在任何软件架构设计中，关于时间和空间的折衷无所不在。任何算法都将占据冯·诺伊曼机器中的空间，并在一定时间内完成。但是，正因为它存在地实在太普遍了，所谓‘静听不闻雷霆之声，熟视不睹泰山之形’，却常常会被人忽略。

事实上，架构设计师不断地在设计过程中作出在“时间”和“空间”间的权衡和取舍。这样的例子随手就可以举出一把，例如，加入预读模块保证游戏地图卷动平滑是电脑游戏常用的伎俩——牺牲空间换取时间；而在存储空间无比金贵手机游戏领域，却常常要牺牲时间换取空间——例如无所不用其极地压缩图片等资源，在读取时解压显示。

于是，架构设计师开始权衡，如哲学家那般探讨时间与空间的本义。

Trade-off 2：Elegancy or Efficiency

不知道你注意了没有，大部分关于架构设计的书中的例子都是使用Java语言。为什么？原因很简单，从面向对象设计的优雅程度上说，鲜有语言能够与Java匹敌——C++连一个单根的类库结构都做不到呢——用Java语言来作为面向对象设计的案例实在是再好不过了。但是，作为一个系统程序员，我是C++语言的忠实拥趸，在这Java横行的年代



图：圣家大教堂照片

窃以不懂Java为豪——因为我所从事的大部分项目都是底层软件，我写下的每一行代码生成的汇编语言序列都可能在一秒中内被CPU反复执行几十万次——哪怕最小的一点性能损失由于累积效应都可能严重影响上层应用软件的性能。在这种情况下，我可是常常连一个虚函数调用的开销都舍不得啊！在没有虚函数的世界里，只有丑陋的函数指针陪伴着我度过漫漫长夜。因为，在我眼里除了效率，还是效率。

在这一点上，我比较欣赏微软关于C#的设计，是的，很多Java支持者批评它在很多地方和Java惊人的相似，作为后来者我觉得这无可厚非——Java大部分的关键字还是从C++来的呢——我的重点在于，C#的设计者在优雅和效率间做了一个不同于Java的设计权衡，你注意到C#有unsafe关键字了么？有了它你可以像C语言那样肆意操作内存，当然你应当为此承担所有的责任。嗯，有unsafe关键字的设计确实不优雅，但是它让我们可以做一些优雅的Java所难以企及的事情。

> C#中可以考虑使用“非安全(unsafe)代码”的情况(来源：MSDN)
>
> 1 处理磁盘上的现有结构
>
> 2 涉及内部包含指针的结构的
>
> 3 高级COM或平台调用方案
>
> 4 性能关键代码

通常而言，“优雅”的架构设计通常设计粒度小、交互关系复杂，从而导致完成同样的功能执行的开销(overhead)不可避免地增多。同时，面向对象设计依赖的语言特性本身就会带来额外开销。仅就面向对象设计的基石——虚函数而言，每个虚函数调用事实上被编译为类似于(*ptr->vptr[3])(ptr)的代码，相比于普通调用，增加了两次寻址的操作(一次寻址虚函数表、一次寻址函数入口)，这一开销在编写系统底层软件时常常是不能忽略的；此外，引入虚函数意味着每个使用到虚函数的类都需要产生虚函数表，在一个庞大的类库体系中，由于类的数量成百上千，这一运行时的内存开销将会十分严重，这便是为什么MFC、wxWidgets等C++类库要用丑陋难懂的宏机制来实现虚函数的原因。

于是，架构设计师又开始权衡，叹了口气，在精巧设计的镂空景泰琅花瓶底下放了几块板砖。

Trade-off 3：Fashion or Familiarity

在摩尔定律作用的今天，对于新技术永远保持好奇是每一个IT人基本的职业素养。只要翻一翻《程序员》每年的合订本，每期上面总会有些新名词出现，而每种新技术看起来总有许多诱人的新特性——尽管技术的更迭其实比娱乐圈的新面孔还快。

所谓“条条大路通罗马”，同样一个软件功能，可以采用不同的技术，从而有许多的实现方案。且不论这些技术本身的优劣、以及取得技术的成本，软件架构师面临来自开发团队自身的挑战——那就是团队本身掌握的技术功底和熟悉程度在很大程度上将制约着软件架构设计中具体技术方案的取舍。

举个最简单但是十分普遍的例子，不管做什么软件都会碰到储存配置信息的问题，如果选择把配置信息保存到文件一定会遇到一个问题——软件的配置文件应当用什么格式？都21世纪了，再用ini恐怕太老土了，还是用XML吧！但是项目组里会XML编程的人又没有，可是偏偏这个软件的配置内容又十分繁杂，如果用ini的话结构会变得很混乱，而且不能利用DTD或者XML Schema等技术来做合法性检验。那么，摆在架构设计师面前的问题便是，硬着头皮用异常复杂的ini描述，还是指派一个家伙去啃《21天学会XML》呢？嗯，这是个问题。前者，意味着难以维护而且易于出错的配置文件，而后者，意味着项目周期可能会延长，而且存在风险——谁知道那个家伙能不能21天搞定XML呢。

于是，架构设计师又开始权衡，晚上的约会，到底是穿卡脚的水晶鞋还是舒服的千层底去好呢？

Trade-off 4：Revolution or Evolution

没有人喜欢收拾烂摊子，可是我们总是会碰到烂摊子——谁让计算机技术发展的快，前年才交付运行的系统，今年已经被视作“历史遗留系统(Legacy System)”——当然，话说回来，要不是IT技术发展如此之快，客户需求层出不穷，我们这些人还不全得画饼充饥去。

历史遗留系统留给架构设计师的挑战是全方位的：原有的硬件设备能否重用？全部重用还是部分重用？原来的数据如何继承？是导入到新系统还是构建一个数据适配层？原有的代码是否有利用价值？最终用户的使用习惯如何延续？林林总总而且往往细枝末节到鸡毛蒜皮地步的问题往往让软件架构师在遗留系统的分析上花去比新系统更多的精力和时间。于是常常有架构师怒不可遏地拍案而起——“原有的垃圾系统全部推翻，老子从新开始！”哇，我可以用最时兴的技术，最高效的开发工具，新系统的架构将如古巴比伦空中花园般优雅，运行将如刘翔跨栏夺冠般快捷，安全性将如万里长城般坚固——这简直是神话中才有的软件——可是，你真的可以这样做么？客户的现有投资需要保护；客户的历史数据至关紧要；不懂技术的大老板觉得从头做起肯定成本要高的多——随便哪一条理由都可以让你心头方才燃起的“革命”热情灰飞烟灭——死心吧，收拾你的烂摊子去。

于是，架构设计师又开始权衡，一边

是历史遗留系统的泥塘，一边是茫然的前路。

Trade-off 5: Under-Design or Over-Design

事实上，关于软件架构设计的本身，也是架构设计取舍的目标，即从粒度上说，软件架构设计需要在不同模块分别细化到何种程度，从时间上说，这一阶段需要进行多长时间。这些因素，架构设计师都要加以权衡。

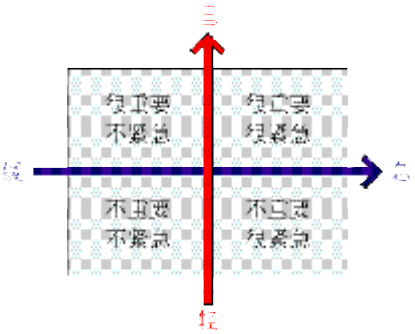
通常而言，软件架构设计的越仔细周到，软件的结构将会越清晰，其扩展性、可维护性等相对较好，但是如果走向极端，往往反而会导致项目失败。首先，架构设计投入人力物力过多将会导致软件开发周期延长、成本上升，而软件的上市时间（time-to-market）推迟，等到推出市场的时候，也许你的竞争对手早就已经占据了半壁河山；其次，从技术层面说，且不论设计粒度过小导致的效率影响，单就80-20法则而言，任何一个系统只有20%的部分是其要害，架构设计也应当服从这一法则，将主要精力用于改善关键部分的结构，而如果对次要部分投入太多，既不经济——增加成本，也不实惠——收效不大。完美主义者架构师对软件结构的完美有着近乎偏执着的追求，但这却是作为工程项目而非艺术设计的软件项目所不允许的，诚然TeX系统或者C++ STL库拥有近乎无懈可击的设计质量，但你的项目可以向它们一样延续数十年而不考虑人力财力么？

Over-Design不好，Under-Design也不好，市场上许多的软件只出了一个版本就销声匿迹，为什么？且不考虑那些市场定位不当的失败品，单就那些市场反响不错确无以为继的软件而言，往往是由于过分强调上市时间（time-to-market），或是由于架构设计师经验、水平的不足，草草甚至胡乱就做出架构上的决策，甚至干脆没有架构设计，导致软件结构破碎混乱，代码复用性差，又没有扩展性可言，要想出新版本只有全部从头再来。遗憾的是，很多好的软件创意初露头角就胎死腹中了。曾经听到一种悲观的说法"中国的软件业做的都是项目，而不是产品"，虽然绝对，但是不无道理，我们缺少的就是从1.0,2.0,3.0直至12.0,13.0的优秀软件产品，而这其中相当程度便是因为架构设计上的Under-Design造成的。

于是，架构设计师又开始权衡，一如城墙上彷徨的哈姆雷特——"To design or not to design, it is a problem."

1978年McCall等定义了11种软件质量的要素——正确性(Correctness)、可靠性(Reliability)、有效性(Efficiency)、完整性(Integrity)、可用性(Usability)、可维护性(Maintainability)、灵活性(Flexibility)、可测试性(Testability)、可移植性(Portability)、可重用性(Reusability)、可互操作性(Interoperability)，完整刻画了软件质量的诸属性。而这11种属性，都可能成为架构设计师权衡的因素。架构设计的过程，实质是在这些不同的矛盾体之间做出一个权衡。任何质量属性都源于我们对软件的某种期望，但即使最优秀的软件，也不可能同时在每个方面都做到完美。评价架构设计的好坏的依据，无非是评价这些权衡的合理性。而评价的标准，在于"轻重缓急"四个字——优秀的软件，总是将"很重要、很紧急"的软件属性做到最好，而对于"不重要、不紧急"的属性，却不必花太多功夫。

大多数情况下，架构设计上的权衡不是"舍鱼而取熊掌"或是"舍身而取义"的"非黑即白"的选择，而是一种折衷。软件开发的领域、条件、环境不同，现有的技术、客户的要求不同，在黑白两个极端之间的张力亦随之变化。软件架构师的使命便是应对这种张力，通过对不同软件质量要素的权衡寻求软件系统设计的平衡点（Trade-off）。一方面，这要求架构设计师具备对系统分析、对客户需求的宏观把握能力，具备对软件结构、对技术方案的微观操作能力，这是架构设计中理性的一面，或者说技术的部分。而另一方面，架构设计作为一种非线性问题，如同Donald Knuth所坚称的那样，架构师在设计过程中应像画家创作新作品般，秉承审美的精神与艺术的态度，对自己的作品充满母亲对孩子般与生俱来的感情和无可推卸的责任感。



从这个意义上说，架构设计是运筹学上典型的"非线性多目标规划"的问题。当一位软件架构师开始他的设计的时候，他的"目标"有很多：可扩展性、可维护性、可测试性、空间效率、时间效率等等——这些"目标"常常互不兼容，甚至完全矛盾！而他面临的"限制条件"也不少，项目周期、团队人手、人员技术积累等，无一不在掣肘着架构设计师，令他不可率性尽情挥洒。"架构设计师"听起来是个"透着金领味"的头衔，可是悲观的说，他们无非是这样一群人：一面为客户收拾着历史遗留系统的烂摊子，一面抱怨项目组里程序员技术水平太差，一面抱怨老板买不起先进的设计工具却只懂一味死催项目进度——架构设计师正是艰难地在这些夹缝中不断作出一个又一个权衡。

作者简介：
柯化成，浙江大学计算机学院硕士研究生，Microsft MVP(Visual C++)。有5年软件开发经验，使用C++和Python语言，专注于架构设计、虚拟机技术。

■ 责任编辑：欧阳璟（ouyangjing@csdn.net）

C# 编码标准

指导和最佳实践

■ 文 / Juval Lowy

编者按

在连续了几期对技术趋势的分析、解决方案的回顾和开发框架的评析之后，我们先后收到了一些读者对于技术专题的反馈。在肯定了前面内容的同时，也希望能够介绍一些面向一线开发人员的“案头手册”。针对这个要求，本期《程序员》的技术专题为C#开发人员组织了一套“行为规范”，希望该规范可以使不同阶层的C#开发人员都能够在时间、效率以及质量上有所收益。.NET技术大师Juval Lowy在2003年发布了该规范的1.0版本。该规范一经发布，便迅速得到了广大C#社区的认同。如今，Juval发布了该标准的2.0版，并将它附加在他最近的新书《Programming .NET Components》(http://www.oreilly.com/catalog/pnetcomp2/index.html) 的附录中。内容涵盖了进行项目最佳实践的方方面面，例如命名规范、编码实践、项目设置和项目结构、.NETframework技术等等。看着这些经过了时间磨练的宝贵经验，《程序员》真诚地希望能和国内广大的C#开发者们一起分享其中的精妙。于是我们对该标准进行了全文翻译，并收集多方的意见，挑选了一些我们认为其中精华的部分，并邀请了国内的技术专家对其进行分析和点评。在此，我们对向本专题提供意见的技术专家们表示感谢。如果其中有不足的地方，也希望广大读者能够提出自己的意见。

命名规范

1.利用Pascal的方式定义类型、方法名和常量

```
public class SomeClass
{
        const int DefaultSize = 100;
        public SomeMethod();
        {}
}
```

2.对于局部变量和方法的参数使用骆驼命名法

```
int number;
void MyMethod(int someNumber)
{}
```

3.接口的名称前加上I

```
interface IMyInterface
{…}
```

4.在私有成员变量前面加上m_。对于m_后面的变量名使用骆驼命名法

```
public class SomeClass
{
        private int m_Number;
}
```

5.对自定义的属性类加上后缀Attribute

6.对自定义的异常类加上后缀Exception

7.方法的命名使用动词——对象对，例如ShowDialog()

8.有返回值的方法的命名中要有对返回值的描述，例如GetObjectState()

9.使用带有说明性的变量名

a)避免单字符的变量名，例如i或t等。使用类似于index或temp这样有意义的名字。

b)对于public或protected类型的变量避免使用匈牙利表示法

c)不要缩写单词（例如用num取代number）

10.总是使用C# 预定义的类型而不要使用在System名称空间中的别名，例如：

使用object而不是Object

使用string而不是String

使用int而不是Int32

11.在使用泛型的时候，类型的首字母要大写。当处理.NET中的Type类型的时候，保留Type后缀。（C# 2.0新特性）

```
//正确
public class LinkedList<K, T>
{…}
//避免
public class LinkedList<KeyType, DataType>
{…}
```

12.使用有意义的名字定义名称空间，例如产品名或者公司名

13.避免通过全限定方式使用类型名称，使用using关键字。

14.避免在一个名称空间中使用using关键字

15.把所有系统框架提供的名称空间组织到一起，把第三方提供的名称空间放到系统名称空间的下面

```
using System;
using System.Collection.Generic;
using System.ComponentModel;
using System.Data;
using MyCompany;
using MyControls;
```

16.使用代理推导而不要显式的实例化一个代理（C# 2.0新特性）

```
delegate void SomeDelegate();
public void SomeMethod()
{…}
SomeDelegate someDelegate = SomeMethod;
```

17.维护严格的代码缩进。不要使用tabs或非标准的缩进，例如一个空格。推荐的缩进是3到4个空格。

18.在和你的代码缩进处于同一个级别处为该行代码添加注释。

19.所有的注释都应该通过拼写检查。注释中的错误拼写意味着开发进度的延缓。

20.所有的类成员变量应该被声明在类的顶部，并用一个空行把它们和方法以及属性的声明区分开

```
public class MyClass
{
    int m_Number;
    string m_Name;
    public void SomeMethod1();
    public void SomeMethod2();
}
```

21.在最靠近一个局部变量被使用的地方声明该局部变量。

22.一个文件名应该能够反映它所对应的类名

23.当使用一个部分类并把该类分布到不同的文件中时，在每一个文件名末尾都加上该文件实现的部分在类整体中扮演的作用。例如：

```
//In MyClass.cs
public partial class MyClass
{…}
//In MyClass.Designer.cs
public partial class MyClass
{…}
```

24.总是要把花括号“{”放在新的一行

编码实践

1.避免在同一个文件中放置多个类

2.一个文件应该只向在一个名称空间内定义类型。避免在一个文件中使用多个名称空间

3.避免在一个文件内写多于500行的代码（机器自动生成的代码除外）

4.避免写超过25行代码的方法

5.避免写超过5个参数的方法。如果要传递多个参数，使用结构。

6.一行不要超过80个字符

7.不要手动去修改任何机器生成的代码

a)如果修改了机器生成的代码，修改你的编码方式来适应这个编码标准

b)尽可能使用partial classes特性，以提高可维护性；（C# 2.0新特性）

8.避免对那些很直观的内容作注释。代码本身应该能够解释其自身的含义。由可读的变量名和方法名构成的优质代码应该不需要注释。

9.注释应该只说明操作的一些前提假设、算法的内部信息等内容。

10.避免对方法进行注释

a)使用充足的外部文档对API进行说明

b)只有对那些其他开发者的提示信息才有必要放到方法级的注释中来

11.除了0和1，绝对不要对数值进行硬编码，通过声明一个常量来代替该数值

12.只对那些亘古不变的数值使用const关键字，例如一周的天数。

13.避免对只读(read-only)的变量使用const关键字。在这种情况下，直接使用readonly关键字

```
public class MyClass
{
    public const int DaysInWeek = 7;
    public readonly int Number;
    public MyClass(int someValue)
    {
        Number = someValue;
    }
}
```

14.对每一个假设进行断言。平均起来，每5行应有一个断言。

```
using System.Diagnostics;

object GetObject()
{…}

object someObject = GetObject();
Debug.Assert(someObject != null);
```

15.每一行代码都应该以白盒测试的方式进行审读。

16.只捕捉那些你自己能够显式处理的异常。

17.如果在catch语句块中需要抛出一场，则只抛出该catch所捕捉到的异常（或基于该异常而创建的其他异常），这样可以维护原始错误所在的堆栈位置。

```
catch(Exception exception)
{
        MessageBox.Show(exception.Message);
        throw; //或 throw exception;
}
```

18.避免利用返回值作为函数的错误代码。

19.避免自定义异常类。

20.当自定义异常类的时候：

a)让你自定义的异常类从Exception类继承

b)提供自定义的串行化机制

21.避免在一个程序集中(assembly)中定义多个Main()方法。

22.只把那些绝对需要的方法定义成public，而其它的方法定义成internal。

23.避免friend assemblies，因为这会增加程序集之间的耦合性。

24.避免让你的代码依赖于运行在某个特定地方的程序集。

25.在application assembly（EXE client assemblies）中最小化代码量。使用类库来包含业务逻辑。

26.避免显式指定枚举的值

```
//正确
public enum Color
{
        Red, Green, Blue
}
//避免
public enum Color
{
        Red=1, Green=2, Blue=3
}
```

27.避免为枚举指定一个类型

```
//避免
public enum Color : long
{
        Red, Green, Blue
}
```

28.对于if语句，总使用一对{}把下面的语句块包含起来，哪怕只有一条语句也是如此。

29.避免使用三元条件操作符。

30.避免利用函数返回的Boolean值作为条件语句。把返回值赋给一个局部变量，然后再检测。

```
bool IsEverythingOK()
{…}
//避免
if(IsEverythingOK())
{…}
//正确
bool ok = IsEverythingOK();
if(ok)
{…}
```

31.总是使用以零为基数的数组。

32.总是使用一个for循环显式的初始化一个引用成员的数组:

```
public class MyClass
{}
const int ArraySize = 100;
MyClass[] array = new MyClass[ArraySize];
for(int index=0; index<array.length; index++)
{
        array[index] = new MyClass();
}
```

33.使用属性来替代public或protected类型的成员变量。

34.不要使用继承下来的new操作符，使用override关键字覆写new的实现。

35.在一个非密封(non-sealed)类中，总是把那些public和protected的方法定义成virtual。

36.除非为了和其它语言进行互动，否则绝不要使用不安全（unsafe）的代码

37.避免显示类型转换。使用as关键字安全的转换到另一个类型。

```
Dog dog = new GermanShepherd();
GermanShepherd shepherd = dog as GermanShepherd;
if(shepherd != null)
{…}
```

38.在调用一个代理前，总是检查它是否为null。

39.不要提供public的事件成员变量。改用Event Accessor。

```
public class MyPublisher
{
        MyDelegate m_SomeEvent;
        public event MyDelegate SomeEvent
        {
                add
```

```
        {
                m_SomeEvent += value;
        }
        remove
        {
                m_SomeEvent -=value;
        }
}
```

40.避免定义事件处理代理。使用EventHandler<T>或者GenericEventHandler。其中GenericEventHandler定义在《Programming .NET components》2/e第6章中。

41.避免显示触发事件。使用EventsHelper安全的发布事件。EventHelper定义在《Programming .NET components》2/e第6～8章中。

42.总是使用接口。

43.接口和类中方法和属性的比应该在2:1左右

44.避免只有一个成员的接口。

45.努力保证一个接口有3~5个成员。

46.不要让一个接口中成员的数量超过20，而12则是更为实际的限制。

47.避免在接口中包含事件。

48.当使用抽象类的时候，提供一个接口。

49.在类继承结构中暴露接口。

50.推荐使用显式接口实现。

51.从来不要假设一个类型支持某个接口。在使用前总是要询问一下。

```
SomeType obj1;
IMyInterface obj2;

/* Some code to initialize obj1, then: */
obj2 = obj1 as IMyInterface;
if(obj2 != null)
{
        Obj2.Method1();
}
else
{
        //Handle error in expected interface
}
```

52.不要硬编码向用户显示的字符串。要使用资源。

53.不要硬编码那些可能会随发布环境变化而变化的字符串，例如数据库连接字符串。

54.使用String.Empty取代“ ”

```
//避免
string name = " ";

//正确
string name = String.Empty;
```

55.使用一个长字符串的时候，使用StringBuilder代替string

56.避免在结构中提供方法

a)参数化的构造函数是鼓励使用的

b)可以重载运算符

57.当声明了静态成员的时候，总是要提供一个静态构造函数

58.当早绑定（early-binding）可能的时候就尽量不要使用迟绑定（late-bingding）

59.让你的应用程序支持跟踪和日志

60.除了要在switch语句块中实现代码跳转，不要使用goto关键字

61.总在switch语句的default情形提供一个断言

```
int number = SomeMethod();
switch(number)
{
        case 1:
                Trace.WriteLine("Case 1:");
                break;
        case 2:
                Trace.WriteLine("Case 2:");
                break;
        default:
                Debug.Assert(false);
                break;
}
```

62.除了在一个构造函数中调用其它的构造函数之外，不要使用this关键字

```
//Example of proper use of 'this'
public class MyClass
{
        public MyClass(string message) {
        }
        public MyClass() : this("Hello") {
        }
}
```

63.不要使用base关键字访问基类的成员，除非你在调用一个基类构造函数的时候要决议一个子类的名称冲突

```
//Example of proper use of 'base'
public class Dog
{
        public Dog(string name)
        {
        }
        virtual public void Bark(int howLong)
        {
        }
}
public class GermanShepherd : Dog
{
        public GermanShepherd(string name) : base(name)
        {
        }
        override public void Bark(int howLong)
```

```
        {
                base.Bark(howLong);
        }
}
```

64.不要使用GC.AddMemoryPressure()

65.不要依赖HandleCollector

66.基于《Programming .NET components》2/e中第4章的内容实现Dispose()和Finalize()方法。

67.总是在unchecked状态下运行代码（出于性能的原因），但是为了防止溢出或下溢操作，要果断地使用checked模式。

```
int CalcPower(int number, int power)
{
        int result = 1;
        for(int count=1; count<=power; count++)
        {
                checked
                {
                        result *= number;
                }
        }

        return result;
}
```

68.使用条件方法来取代显式进行方法调用排除的代码（#if…#endif）

```
public class MyClass
{
        [Conditional(" MySpecialCondition ")]
        public void MyMethod()
        {}
}
```

69.不要在泛型接口中定义约束。接口级的约束通常可以利用强类型来替代

```
public class Customer
{…}
//避免:
public interface IList<T> where T : Customer
{…}
//正确:
public interface ICustomerList : IList<Customer>
{…}
```

70.不要在接口上定义方法相关的约束。

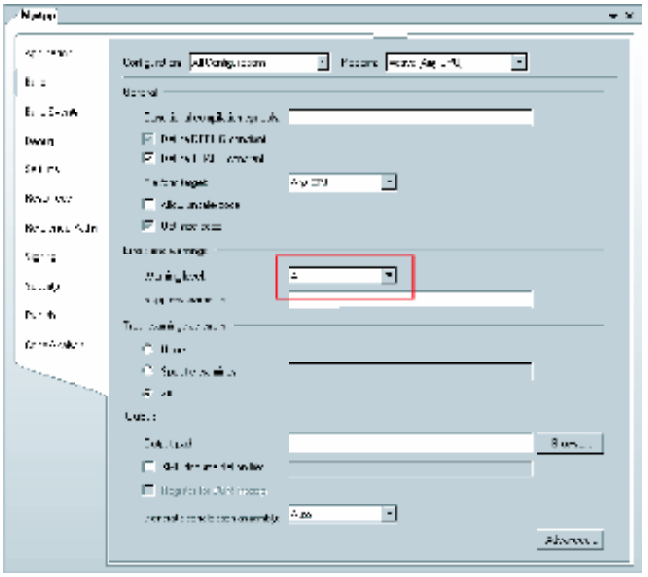
71.不要在代理上定义约束。

72.如果一个类或方法提供了泛型和非泛型版本，那么优先选择泛型版本

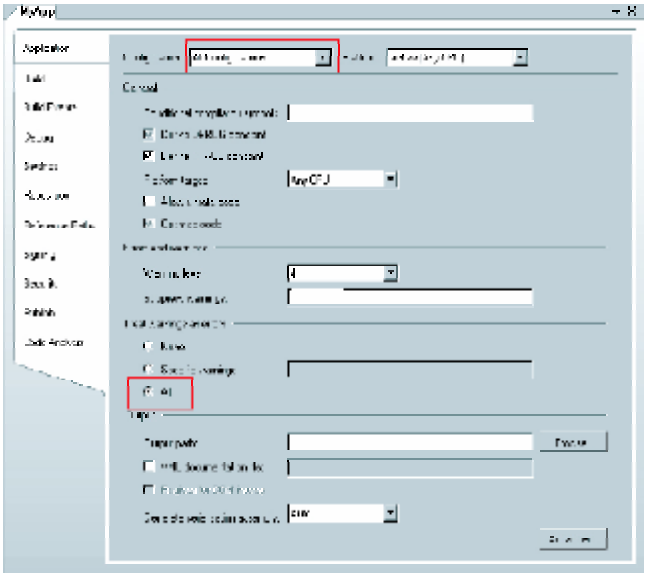
项目设置和项目结构

1.总是在4级警告上建立你的项目

2.在发布版（Release）中，把警告当成错误来对待



（注意这并不是Visual Studio的默认选项）。我们建议在调试版（Debug）中也这样设置，尽管这不是强制的要求。



3.避免关闭编译器的某些警告选项

4.总是要在应用程序的配置文件中显式指定支持的运行时版本

```
<?xml version="1.0"?>
<configuration>
        <startup>
                <supportedRuntime version="v2.0.5500.0" />
                <supportedRuntime version="v1.1.5000.0" />
        </startup>
</configuration>
```

5.避免显式进行CLR程序集版本的重定向和绑定。

6.避免显示的预处理定义（#define）。使用项目设置来定义条件编译常量。

7.不要在AssemblyInfo.cs中加入任何逻辑。

8.不要在AssemblyInfo.cs之外的文件中添加程序集属性。

9.提供AssemblyInfo.cs中的所有信息，例如公司名，描述和版权事项等。

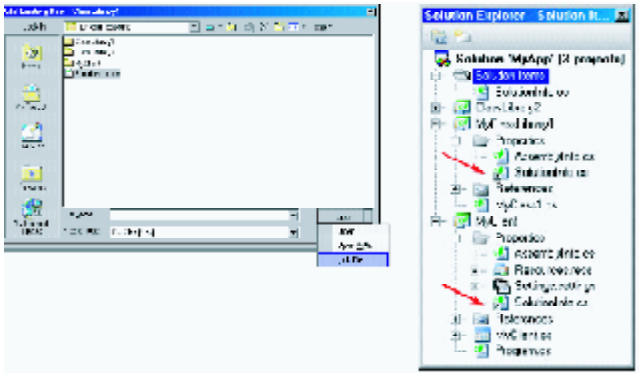
10.同一个解决方案中的程序集引用都应该使用相对路径。

11.禁止在程序集之间出现循环引用。

12.避免多模块（multi-module）程序集。

13.避免利用异常窗口（调试|异常）削弱异常处理。

14.坚持在同一个解决方案中的所有程序集之间使用统一的版本号。

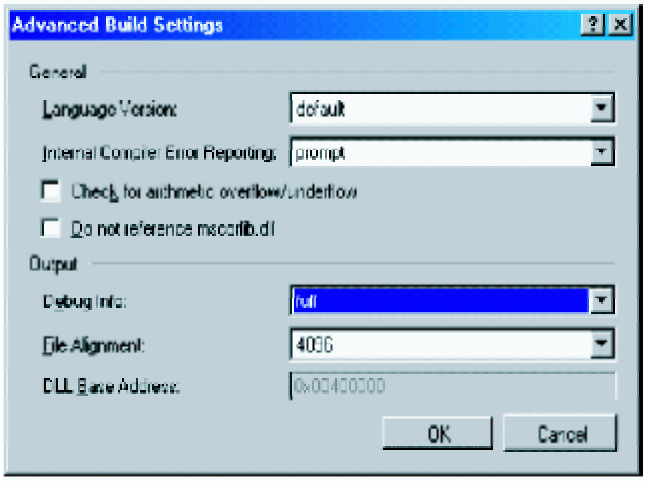


15.把所有解决方案的信息存放到一个共享的SolutionInfo.cs文件中。

16.把你的应用程的配置文件命名成App.config，并把它包含在你的项目中。

17.修改Visual Studio 2005的默认项目结构来适应你的项目规划，并且对项目文件夹和文件使用统一的结构。

18.一个Release发布应该包含调试信息



19.总是对你的程序集进行签名，包括你的客户应用程序

20.使用密码保护的键.NET Framework相关的指南

.NET Framework相关的指南

数据访问（评析人：邹建峰）

1.总是使用强类型DataSet。避免直接使用ADO.NET

评析：强类型DataSet是从DataSet类派生的类，除了有DataSet类的方法和属性外，还具有数据库中所有对应的对象的属性和信息，它可以提供以名字为索引的方法来直接访问表和列，使用强类型DataSet不仅可以提高代码的可读性，还可以利用编译器在编译时就检查出类型不匹配错误，可以提高代码的一致性、可读性和可维护性，从而可以避免使用ADO.NET中在运行才能检查出来的诸多错误。

2.当访问数据库的时候总是使用事务

a)总是使用企业服务或System.Transactions事务

b)不要使用ADO.NET事务by enlisting the database explicitly

3.总是将事务隔离级别（transaction isolation level）设定为Serializable。如果要设定为其他的级别，则需要有实际业务管理规则支持。

4.不要在Windows Form、ASP.NET Web Form或者Web Service程序中使用Data Source窗口来丢弃数据库连接，这样会使表现层与数据层耦合。

评析：诚然，在Visual Studio.net开发工具中使用拖放的方式，将数据库连接控件放在界面上是一件十分方便的事情，但是这种方面有很多的缺陷，比如连接串是固定不变的，控件的生存期是整个页面的生命期，每次post后还会生成相应的控件（大多数情况下，在Post后，无需要再次新建数据库连接），此外，对分层的开发来讲，界面层的改动比数据层要相对频繁，因此，数据库连接融入用户界面中，导致整个应用程序的代码维护更加困难。

5.使用Windows身份认证，不要使用SQL Server认证

评析：使用Windows身份认证，可以结合Windows服务器的强大的域管理机制，使得数据库管理和域管理紧密的结合在一起。Windows身份认证比SQL Server认证提供更多的特性，容易管理和使用。此外，在其连接串中也无须将密码以明文形式放在其中。因此，除非是服务器不能访问域控制器或特殊情况，不然就应该使用Windows认证。

6.访问SQL Server的组件应当与发起调用客户程序运行在不同的id之下。

评析：这点是很明显的，不然的话，客户程序被修改的话，就很可能导致数据的紊乱。

7.总是用高级的、类型安全的类来封装你的存储过程。只允许通过该类来访问存储过程。使用Visual Studio 2005中类型安全的数据适配器来尽可能自动完成这一工作。

8.避免在存储过程中加入逻辑。如果在存储过程中出现一个IF，那么你就犯错误了。

评析：一般来讲，存储过程只是封装了数据库的访问，从逻辑的易变性来看，不应该将业务逻辑都封装到存储过程中，因为在程序运行中修改存储过程几乎是不现实的，将运行的参数传入存储过程中运算也会导致整个应用的性能下降。但是这并不代表在存储过程不能出现IF，因为在数据层面的必要的判断还是需要的，比如判断记录是否存在来决定是否是插入一条记录还是修改记录，这整个过程都封装到存储过程中是合理的

ASP.NET 和 Web Services（评析人：金戈）

1. 避免在ASP.NET中的ASPX文件中放入代码。所有的代码都应该放到code-beside partial class中。

2.在ASP.NET的code-beside partial class中的代码应该调用其它组件而不是直接包含业务逻辑。

3.在访问一个Session变量前，总要检查它是否为null。

4.在事务性页面或Web Services中，总是将Session保存在SQL Server中。

5.避免把ASP.NET的服务器控件的Auto-Postback属性设置为true

6.打开ASP.NET页面的智能浏览(Smart Navigation)

7.尽力为web services提供接口。详见《Programming .NET Components》2/e。

8.总是为web services提供名称空间和服务描述

9.总是对web方法提供描述

10.当添加一个web service引用的时候，为位置提供一个有意义的名字

11.在ASP.NET页面和web service中，用一个局部属性来封装一个Session变量，只有这个局部属性有全访问Session，其他的代码只能访问这个属性，而不能直接访问Session变量。

```
public class Calculator : WebService
{
        int Memory
        {
                get
                {
                        int memory  = 0;
                        object state = Session[" Memory "];
                        if(state != null)
                        {
                                memory = (int) state;
                        }
                        return memory;
                }
                set
                {
                        Session[" Memory "] = value;
                }
        }
        [WebMethod(EnableSession=true)]
        public void MemoryReset()
        {
                Memory = 0;
        }
}
```

12.总是要修改客户端的web service封装类以支持cookies，因为你并不知道服务是否使用会话状态。

```
public class Calculator : SoapHttpClientProtocol
{
        public Calculator()
        {
                CookieContainer = new System.Net.CookieContainer();
                Url = …;
        }
}
```

评析：

1 合理的分布代码

规范中的前两条都是关于代码逻辑结构的问题。逻辑结构问题通常并不影响代码的执行结果，但对代码的设计、开发和维护都有很大的意义。举个例子，没有面向对象技术的时候，一样可以写出程序，而有了面向对象编程技术之后，复杂代码的结构更加合理化了，也同样可以完成以前不是面向对象代码所能实现的功能。

在ASP.NET的最初设计中，微软引入了一种叫“code-behind”的逻辑代码结构（即ASPX对应的类派生自code-behind类），用于清晰的将表现代码和业务逻辑代码分离，形成更容易开发和维护的代码组织结构。在C# 2.0里，语言引入了“partial class”（部分类）的语法，同时ASP.NET 2.0也将这种code-behind的逻辑结构进一步演变为code-beside模型（即

ASPX对应的类以部分类的形式作为code - beside类的补充部分）——这些变化都是为了形成更好的代码逻辑结构。

2 严谨的编码习惯

规范中的第3条和第11条都是关于养成严谨的编码与开发习惯的。

首先，使用任何引用类型的数值之前，均应该检查该值是否为null，否则NullException就离你不远了。因为Session是一个返回值为object的元素属性（item property），而object是引用类型的鼻祖，那么显然要提防返回null的情形。比如经常可以看到的这样的写法：

```
int lastItemID = (int)Session["LastItemID"];
```

一旦在某些特定的场景中取不到LastItemID这个会话变量的话，马上就会产生空引用异常。因此，比较严谨的写法应该是：

```
object sessionLastItemID = Session["LastItemID"];

int lastItemID = -1; // 取一个正常情况下不会出现的值做默认值

if (sessionLastItemID != null)
{
    lastItemID = (int)sessionLastItemID;
}
```

之后就可以放心使用lastItemID这个值了。更进一步，我们可以用属性将对这个会话变量的访问封装起来，就像这样：

```
public int LastItemID
{
    get
    {
        object sessionLastItemID = Session["LastItemID"];
        // 取一个正常情况下不会出现的值做默认值
        int lastItemID = -1;
        if (sessionLastItemID != null)
        {
            lastItemID = (int)sessionLastItemID;
        }

        return lastItemID;
    }
    set
    {
        Session["LastItemID"] = value==null ? -1 : value;
    }
}
```

这样做有几个好处。首先，属性的类型定义反映了会话变量本质的类型，以强类型来使用这些变量可以让代码更清晰、更健壮；第二，利用属性访问器隔离会话变量的存储机制，可以有效地隔离这个易变的因素，在数据存储机制发生变化时可以控制代码变化的范围，减少出错的机会；最后，由于使用了属性访问器，可以在访问会话变量的时候加入灵活的业务逻辑处理，比如说对默认值的处理、简单的加密、解密和变换等等。

总之，看上去非常简单的两条规范，可以帮助我们形成良好的编码习惯。

3 以人为本的开发意识

我们在编码标准的培训中常挂在嘴边的一句话是，代码首先是写给人看的，然后才是写给计算机看。所谓写给人看，既是指写给自己看，也是指写给可能接触到代码的其他人。

由此推论，不光代码本身要给后来的维护人员看，程序中产生的出错信息也要考虑给负责软件运行和维护的管理员看，Web Service及其中的Web Method也都要让引用它的开发人员看明白。

因此就不难理解这几条的意义了。看一个MSDN上的实例就明白该如何做了：

```
[WebService(Description="blah blah blah", Namespace="uri:something")]
public class ServerVariables: WebService
{

    [WebMethod(Description="call me for free…")]
    public string GetMachineName()
    {
        return Server.MachineName;
    }
}
```

添加进来的Web Service Reference的位置也需要命名一下，道理是一样的，因为这个名字也是代码的一个组成部分，当一个含义模糊的名称出现在代码中的时候，肯定会让代码晦涩难读，也就大大减低了代码的可维护性。

4.3 多线程 （评析人：崔轶）

1.使用同步域（Synchronization Domains）。详见《Programming .NET Components》2/e。要避免人工同步，因为这经常会带来死锁或者竞争条件。

2.绝对不要调用你的同步域外面的内容。

3.通过一个回调函数管理asynchronous call completion。不要等待，投票或者阻塞竞争。

4.总是为你的线程命名。名称会显示在调试器线程窗口中，这可以使调试过程更加有效率。

```
Thread currentThread = Thread.CurrentThread;
string threadName = "Main UI Thread";
currentThread.Name = threadName;
```

5.不要在一个线程上调用Suspend()或Resume()

6.除非在下列条件下，否则不要调用Thread.Sleep()

a)用Thread.Sleep(0)强制上下文切换是一个可允许的优化技术

b)测试模拟代码中可以使用Thread.Sleep()。

7.不要调用Thread.SpinWait()。

8.不要调用Thread.Abort()来结束一个线程。利用同步对象信号使线程结束。详见《Programming .NET Components》2/e 第8章。

9.避免显式设置线程优先级来控制线程的执行。你可以根据任务的重要程度设置线程的优先级，例如使用ThreadPriority.BellowNormal为一个屏幕保护程序设置低于正常的优先级。

10.不要去读取ThreadState属性。使用Thread.IsAlive()来决定线程的生存状况

11.不要通过将线程类型设为后台线程的方式来关闭应用程序，应使用看门狗（Watchdog）或其他监视对象来确定性地关闭线程。

12.除非能够保证线程的亲缘性（Affinity），否则不要使用线程局部存储。

13.不要调用Thread.MemoryBarrier()

14.在调用Thread.Join前确保你并不是正在Join到本线程上。

```
void WaitForThreadToDie(Thread thread)
{
    Debug.Assert(Thread.CurrentThread.GetHashCode() !=
    thread.GetHashCode());
    thread.Join;
}
```

15.总是使用lock()语句而不要显式使用Monitor进行操作

16.总是把lock()封装在它保护的对象内

```
public class MyClass
{
    public void DoSomething()
    {
        lock(this)
        {…}
    }
}
```

17.你可以自己使用同步方法来取代lock()语句

18.当你想对多个对象加锁的时候，不要对它们逐个加锁（详见《Programming .NET Component》2/e第8章）。

19.避免使用Monitor来等待或者激活对象。使用手动或者auto-reset事件来完成这个工作。

20.不要使用易变（volatile）类型的变量。如要确保确定性的、线程安全的访问，应当要锁住你的对象或者字段。不要使用Thread.VolatileRead()，Thread.VolatileWrite()或者volatile修饰符。

21.避免增加线程池中的最大线程数

22.不要使用叠置的lock语句，因为该语句不提供原子性的加锁。应使用WaitHandle.WaitAll()。

```
MyClass obj1 = new MyClass();
MyClass obj2 = new MyClass();
MyClass obj3 = new MyClass();
//不要使用叠置的lock语句
lock(obj1)
lock(obj2)
lock(obj3)
{
    Obj1.DoSomething();
    Obj2.DoSomething();
    Obj3.DoSomething();
}
```

评析：

多线程技术的重点始终都是共享数据的同步，在.NET平台上也是如此。从作者在多线程这部分提出的这些编码标准来看，也充分证实了这个问题。但在.NET的世界里，还有一些同步数据外的问题，中国有句古话叫做入乡随俗，那么身在.NET大环境中，你也要遵守一定的规则。在这里找出了一些比较有指导意义的部分，和大家交流一下。

首先，在考虑如何同步一个对象的时候，你应该先仔细考虑是否能够不使用共享对象。因为，保持线程之间的接触面最小是可以让线程开足马力的重要前提。这在总是把lock()封装在它保护的对象内这一条中有所体现。我想这和"只在需要的时候才使用同步"这句话的含义是相同的。因为在一个对象上开/解锁都会带来一定程度上的系统开销，另外，lock()所涉及的领域越大，就意味着有越多的线程需要等待，从而带来更多的同步开销和潜在错误。从某种意义上来说，lock的含义就是临时禁止应用程序的多线程功能，所以应尽量减少在锁内执行的语句，以最大的发挥多线程的优势。

其次，在.NET的托管环境下，你要尽量控制自己对线程的控制欲望。因为.NET framework为了GC系统地正确运行，会对线程的执行采取一些额外控制，而这些控制是你所无法预期的。这也就是为什么作者提到了不要在一个线程上调用Suspend()或Resume()的原因。在你调用了这两个函数后，线程总会存在着一点"惯性"，而后才会开始如你所愿的停止或运

行，这和你利用Native API来控制线程是不一样的。另外，你也不要对线程的优先级有过多的指望，即不要妄想通过提高自己线程的优先级来让自己的线程崭露头角，因为你不可能为自己的线程制定过高的优先级，这会妨碍系统的整体性能，那么你也就不能保证会有比你优先级更高的线程出现，于是你最好还是打消想借着优先级“加塞”的想法，一般说来，你只需要根据你的任务需要为线程指定一个“指导性”优先级就足够了。这也就是为什么作者会提出“避免显式设置线程优先级来控制线程的执行。你可以根据任务的重要程度设置线程的优先级，例如使用ThreadPriority.BellowNormal为一个屏幕保护程序设置低于正常的优先级”这样的理由了。我们看到，优先级不是一种保障，它更多的体现着你对某个任务语义上的一种描述。我们还可以从作者对于Sleep的说明上看到了对于线程控制的不确定性：

不要调用Sleep，除了下面的条件：

a)用Thread.Sleep(0)强制上下文切换是一个可接受的优化技术

b)在测试或模拟代码中可以接受Thread.Sleep()的实际效果

仔细体会作者提到的可接受的这个词，你就能感受到强烈的不确定性了。

第三，学会让你的线程善始也善终。在.NET这个由丰富的逻辑和规则构成的世界里，忽略任何你认为无用的东西都会给你带来麻烦。也许你会为你的线程设计出很精彩的同步逻辑，也会精细考量每一个线程的创建时机，那是因为你需要它为你完成某些工作。然而，在你不需要它们的时候，你是否仔细考虑过如何终止它们？在传统的win32 API中，TerminateThread可以让你果断的杀死一个线程，干净而利落。那么在.NET中呢？你是否认为Thread.Abort()也可以让你获得快刀斩乱麻的快感呢？答案可能会让你失望，因为Thread.Abort()虽然是Thread类中的一个方法，但是却并不推荐用来结束某个线程。原因有三：

1.这并不是一个很“光彩”的方法。线程在复原之前都会交出自己的资源，并清理自己的“内务”，尽管在调用了Thread.Abort()后，你可以通过捕获ThreadAbortException异常来自己做这些写工作，但是通常异常处理被推荐用来处理unhandled exception的资源回收工作，而并不把它作为结束线程的标准方法。

2.你也无法保证catch代码里有与清理线程无关的操作

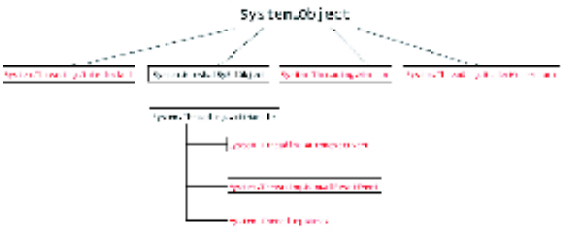
3.如果你的线程使用了COM interop或者P-Invoke的话，那么在这些外部操作执行的时间里，当你希望执行Thread.Abort()时，那么这个函数总会在这些外部操作完成后才会起作用。

基于以上3个原因，你真的无法预期Thread.Abort()可以按照你的希望来工作。其实，出现上面这些问题的原因很简单，我们刚才一直在.NET这个自动化的托管环境中试图自己控制些什么，所以总会处处碰壁。如果我们只是单纯的告诉.NET运行时，什么时候可以做什么事情，然后就全身而退，那么一切就都会变得自然和正确。具体到结束线程这个问题上，我们可以设置一个变量作为信号，然后让线程在执行的时候不断检查这个信号来确定它是否可以退出。而这里所谓的Abort()函数，就是要以原子方式来设定结束信号并为线程的结束工作做好准备（例如等待所有线程引用被释放，释放所有为了同步而设定的资源等），当.NET看到这些都被处理妥当时，就会自动为你结束线程。

最后，想谈一谈和同步方法相关的具体事宜。作者在这里主要提了三方面的内容，第一，如果你经过再三考量一定要使用同步技术，那么.NET提供的自动同步设施肯定是你首先要考虑的；第二，对于要以原子方式执行的代码块，要尽量使用lock语句而不要使用Monitor类；第三，对于特定的同步内容，可以使用自己的同步方法。仔细想想，自动同步设施和lock有何异同？lock语句相比较Monitor有什么好处？特定的同步内容又为何物？作者似乎并没有更近一步说明，这里谈一些自己的感想。先来看第一个问题，这里作者提到的自动同步设施指的是同步域（Synchronization Domain）。它是专门解决编写线程安全的组件而开发的技术。为了利用.NET组件服务，所有的组件都会被.NET赋予一个上下文环境（Context），而编写组件的类必需要从ContextBoundObject继承。当你把Synchronization属性应用到一个基于ContextBoundObject的类的时候，.NET就会在该组件的Context上绑定一个锁，这样就可以自动保证对组件的同步访问了，那么，如果需要对一组组件原子访问，那么为每一个组件绑定一个锁显然是不正确的，一方面效率不彰，资源浪费；另一方面，还会操成死锁。那么不如把所有的组件绑定到一个锁上，效率，安全等问题皆迎刃而解。总之，这是一个组件级线程安全的技术。而不是代码级的。显然组件级的粒度有些大，毕竟线程安全意味着效率上的折扣。于是我们看到了下面的条款和技术。

总是使用lock语句来实现同步而不要显示使用Monitor来操作。这是在源码级进行保护的一个语言设

施。但我个人并不是很赞同这条，因为夹在Monitor.Enter和Monitor.Exit之间的关键区（critical section）代码会被获得锁对象的线程以独占的方式运行，所以，为了避免死锁，你可以把包含关键区的代码放到一个try块中，之后把Monitor.Exit放到finally块中，一旦关键区代码发生异常，被锁定的资源还可以有逃生的机会，整个事情还不至于太糟。但是如果我们把关键区放到用lock包含的{}中呢，一旦发生故障，似乎所有{}内成员皆在劫难逃。在MSDN上清楚的写到，Monitor.Enter和Monitor.Exit组和的功能和C#中的lock关键字是相同的，但对待异常这个问题，显然灵活性和简便性之间你还是要有所取舍。在这一段的开始，我们还提到了关于特定同步内容的问题。作者说：你可以使用自己的同步方法而不使用lock语句。何时要自己动手？我们搜索一下System.Threading名字空间，找到所有的和同步访问有关的类，就不难找到答案。一般来说，我们常用的同步类有如下几个



其中红色显示的是可以被我们利用的同步类，而特定的情况也就源于此：

● Interlocked用于对变量以原子方式进行增量、比较等操作。你只要记住在多线程环境下，所有i++这样的操作都要变成Interlocked::Increment(&n)，其他的操作(i--或==)都要采取类似方式。

● Mutex、AutoResetEvent和ManualResetEvent这三个对象可以算作比较纯粹的同步对象，唯一不同的地方是事件对象可以分为Auto和Manual两类，获得Manual对象的线程要调用Reset把事件对象设定成无信号状态。

● Monitor前面说过了，这可以保证临界区代码按照原子方式执行。

● ReaderWriterLock用于实现“单线程写入，多线程读取的”效果。

最后一个要说的同步方法就是利用属性来约束一个方法。它就是System.Runtime.CompilerServices.MethodImplAttribute只要在方法前面把这个属性指定为Synchronized那么其他问题你就可以放心的交给编译器去做了。

串行化

1.推荐使用二进制的格式

2.把串行化处理过程定义成private方法

3.使用泛型的IGenericFormatter接口。详见《Programming .NET Components》2/e第9章。

4.总是把非密封类标记成可串行化（Serializable）的。

5. 当在一个非密封类中实现IDeserializationCallback接口的时候，要确保其子类可以调用基类的OnDeserialization。详见《Programming .NET Components》2/e第3章。

6.总是要把一个不可串行化（un-serializable）的成员变量明确标记成NonSerializable。

7.总是为一个串行化类（serialized class）中的delegate做上不可串行化标记

```
[Serializable]
public class MyClass
{
        [field:NonSerialized]
        public event EventHandler MyEvent;
}
```

Remoting

1.优先选用管理型配置，而不是可编程配置。

2.总是在single call objects上实现IDisposable接口。

3.如果没有防火墙，当进行remoting的时候优先使用TCP通道和二进制的格式。

4.总是为一个单体对象(singleton object)提供一个null lease：

```
public class MySingleton : MarshalByRefObject
{
        public override object InitializeLifetimeService()
        {
                return null;
        }
}
```

5.总是为由客户端激活的对象提供一个发起人（sponsor）。这个发起人应该提供一个初始的租赁（lease）时间

6.在客户端应用程序关闭的时候，总是要注销（unregister）掉这个发起人

7.总是把远程对象放到类库中

8.避免使用SoapSuds

9.避免在IIS中驻留(hosting)

10.避免使用uni-directional信道。

11.总是在Main()中加载一个远程配置文件，甚至该文件是空的，或者应用程序不需要远程访问的时候也是如此

```
static void Main()
{
    RemotingConfiguration.Configure(" MyApp.exe.config ");
    /* Rest of Main*/
}
```

12.避免使用Activator.GetObject()和Activator.CreateInstance()来激活远程对象，使用new来做这个工作。

13.总是在客户端注册端口0来允许回调

14.总是将客户和主机上的类型过滤提升至完全级别，从而允许回调。

安全 （评析人：李彦智）

1.对于那些就应用来说是private的，而本身又是public的程序集和组件（这样只有你能够使用它们），始终设定你自己的强名称。

```
public class PublicKeys
{
    public const string MyCompany =

  " 12345678948000009400000006020000000240000 " +
  " 525341310004000001000100007D1FA57C4AED9F0 " +
  " A32E84AA0FAEFD0DE9E8FD6AEC8F87FB03766C83 " +
  " 4C99921EB23BE79AD9D5Dcc1DD9AD23613210290 " +
  " 0B723CF980957FC4E177108FC607774F29E8320E " +
  " 92EA05ECE4E821C0A5EFE8F1645C4C0C93C1AB99 " +
  " 285D622CAA652C1DFAD63D745D6F2DE5F17E5EAF " +
  " 0FC4963D261C8A12436518206DC093344D5AD293 ";
}
[StrongNameIdentityPermission(SecurityAction.LinkDemand

PublicKey = PublicKeys.MyCompany)]
public class MyClass
{…}
```

评析：在.NET应用程序中使用强名称，能给应用程序带来许多益处：确保名称唯一性、版本管理、可靠性检查等。通过使用强名称，也易于程序在GAC中的注册与其运行权限的设定。

2.在应用程序配置文件中加入加密和安全保护。

评析：在.NET平台中，为了加强互操作性，大量使用XML，其简单、易用、易理解及其它优点是大家有目共睹的。.NET平台中的应用（如Web应用、WinForms应用都在使用基于XML的设置文件来保存应用程序的配置信息，非常易于使用，于是大家把一些敏感的应用程序配置信息放在了配置文件里，如数据库服务器连接用户名或密码、安全角色配置信息等，这些明文的信息很容易被那些不怀好意的人利用，因此对这些敏感信息对行加密，并对文件本身进行保护，是非常必要的。要知道，在安全防护上多花费一分力气，则黑客可能要花十分或更多的力气才能获取并利用它，这个投入很值得。

3.当导入一个interop方法的时候，对非托管代码的权限进行断言，并采用合适的权限

```
[DllImport(" user32 ",EntryPoint= " MessageBoxA ")]
private static extern int Show(IntPtr handle, string text,
                    string caption, int msgType);
[SecurityPermission(SecurityAction.Assert,
                UnmanagedCode = true)]
[UIPermission(SecurityAction.Demand,
              Window = UIPermissinWindow.SafeTopLevelWindows)]
public static void Show(string text, string caption)
{
    Show(IntPtr.Zero, text, caption, 0);
}
```

评析：非托管代码可能带来诸多安全问题，因此在托管平台上使用非托管代码时，要考虑进行一些声明性安全设置，对其代码执行及用户界面上进行必要的限制，帮助防止潜在的有害代码发起欺骗攻击，例如模仿受信任的系统对话框。

4.不要通过SuppressUnmanagedCodeSecurity属性抑制对非托管代码的访问

评析：使用SuppressUnmanagedCodeSecurity属性后，就禁用SecurityPermission.UnmanagedCod的安全检查，从而可能造成进入非托管代码时，没有进行运行时安全检查，所以要避免这样做。

5.不要使用TlbImp.exe中的/unsafe选项。应在托管代码中包装RCW，这样就可以以声明的方式在wrapper上断言和指定权限。

评析：使用/unsafe选项表示在不进行.NET Framework安全检查的情况下生成接口，因此调用以此方式公开的方法可能会导致安全风险。如果对此方法不够了解，则最好不要使用这个选项。通过在托管代码中包装RCW，可以使用.NET Framework提供的代码声明性安全功能，限制代码的权限，提高代码的安全性。

6.在服务器上，配置一段访问安全策略的代码，该代码只对Microsoft，ECMA和自己（通过强名称鉴定）提

供完全信任。而对于其它地方的代码则不授予任何信任。

评析：服务器一般执行关键的服务程序，因此需要严格加以保护，因此在代码安全策略的限制上，明确信任已知来源的代码，是保证安全的一个最佳实践。

7.在客户端机器上，配置一个安全策略。该策略只允许客户端程序具有执行、调用服务器和向用户显示界面的权限。当没有使用ClickOnce时，客户端应用程序应该在代码组(code groups)进行强名称。

评析：除 ClickOnce 外，客户端的应用程序也应该进行较为严格的安全权限限制，尤其对于企业中系统管理员来说，制定这样一个统一的策略并部署是很有必要的。

8.为了对抗“引诱攻击”(luring attack)，总是在程序集层（Assembly level）拒绝所有和程序运行不相关的权限。

```
[assembly:UIPermission(SecurityAction.RequestRefuse,
Window=UIPermissionWindow.AllWindows)]
```

评析：SecurityAction.RequestRefuse是最严格的设置，可以防止代码会恶意地访问系统资源。

9. 总是在Windows的每一个Main()方法中设定安全上下文的用户对象策略（Principal Policy）：

```
public class MyClass
{
    static void Main()
    {
        AppDomain currentDomain = Thread.GetDomain();
        currentDomain.SetPrincipalPolicy(

        PrincipalPolicy.WindowsPrincipal);
    }
            //other methods
}
```

评析：这是一个很好的编程习惯，让每一个程序都以特定的用户身份运行，有助于利用 .NET 提供的基于角色的安全性功能并保护系统。

10.如果没有指定一个不同的允许权限，则不要对该权限实施断言。

System.Transactions

1.TransactionScope对象必须清理（Dispose）。

2.在一个事物范围内部，在Complete()被调用后不要放置任何代码。

3.不要多次调用 Complete()。

4.设定一个 ambient transaction 时，应保存旧的 ambient transaction，用完之后，应当恢复。

5.在Release版中，不要把事务超时设定成0(超时时限为无穷)

6.当克隆一个事务的时候，总是使用DependentCloneOption.BlockCommitUntilComplete来完成。

7.为每一个工作线程构造一个新的相关克隆，禁止将同一个相关克隆传给多个线程。

8.不要把一个事务的克隆传递给TransactionScope的构造函数

企业服务 （评析人：思归）

评析：

为了响应企业开发的需求，微软随Windows 2000服务器推出了COM+，一个COM运行时环境，提供了包括同步，对象池，基于角色的安全，以及事务在内的组件服务(详见9，10)，以方便开发scalable，secure，transacted的分布式应用。

在.NET之前，COM组件使用这些服务，其机制是这么实现的：开发人员通过基于attribute的declarative 方式设置所需服务，譬如通过Component Services Explorer(CSE)或者COM+catalog管理对象API来更新COM+catalog ，在运行时，COM+则拦截调用组件的方法，根据组件的这些设置，建立一个满足组件需求的运行环境，然后进而调用组件方法。

在.NET 框架下使用这些服务，需要通过System.EnterpriseServices 命名空间下的类来实现。在.NET里，使用COM+服务的 .NET 组件叫做服务组件(其实翻译成“被服务组件”或“受服务组件”更合适些)，需要继承ServicedComponent类。开发人员也可以使用CSE设置服务，但.NET中的做法一般是，开发人员在程序集、类、方法上添加Attribute来设置所需的服务。

一个对象要想使用COM+服务，是要花代价的，除了拦截会影响性能外，系统为需要COM+服务的对象建立一个context，大概每个context需要2-3K的内存(详见1)，所以如果没必要的话，不要把.NET组件做成ServicedComponent。

使用Attribute来设置这些组件服务，极大地简化了编码，使程序员可以更专著于业务逻辑上，有利于提高productivity。而且Attribute的方式，提倡并强化了separation of concerns的思想，把基础设施类的编码与业务编码分离，使得编码容易维护。

这个编码标准的作者强调的是what和how，对why着墨甚少，但想深入理解，对why的认识还是必不可少的，下面对有关企业服务方面的几个编码标准做个粗浅的解释。

1和2是关于Transaction的，3，4和5是关于Instance management的，6-11是关于Security的。

1.不要在一个事物性的方法中捕获异常。使用AutoComplete属性。

2. 不要调用SetComplete()，SetAbort()和类似的函数。使用AutoComplete属性

```
[Transaction]
public class MyComponent : ServicedComponent
{
        [AutoComplete]
        public void MyMethod(long objectIdentifier)
}
```

评析：对设置[Transaction]的组件，系统会根据组件的事务选项，在运行时设置好一个运行环境，提供恰当的事务服务。编码可以通过System.EnterpriseServices命名空间下的ContextUtil类里的诸多静态属性和方法来影响事务的结果，包括MyTransactionVote属性以及SetComplete和SetAbort方法，譬如

```
[Transaction]
public class MyComponent : ServicedComponent
{
        public void MyMethod(long objectIdentifier)
        {
                try
                {
                        //do your work here

                        ContextUtil.SetComplete();

                        //or ContextUtil.MyTransactionVote =
                        //TransactionVote.Commit;

                }
                catch (Exception ex)
                {
                        ContextUtil.SetAbort();

                        //or ContextUtil.MyTransactionVote =
                        //TransactionVote.Abort;
                        throw;
                        // again to propagate the exception if necessary
                }
        }

        //...
}
```

这样的编码，把事务管理和业务逻辑混在一起，违反了separation of concerns的原则，而且需要到处重复类似编码，很明显跟作者的编程标准的思想相违。COM+和Enterprise Services提供了AutoCompleteAttribute，能达到同样的目的，譬如

```
[Transaction]
public class MyComponent : ServicedComponent
{
        [AutoComplete]
        public void MyMethod(long objectIdentifier)
        {
                //....do work
        }
}
```

假如方法正常返回，系统会自动调用SetComplete，假如在方法里有异常抛出，那么事务就会终止。

3.总是覆盖CanBePooled方法并返回true(除非你有理由不返回池中)

```
public class MyComponent : ServicedComponent
{
        protected override bool CanBePooled()
        {
                return true;
        }
}
```

4.总是为一个对象池中的对象（pooled objects）显式调用Dispose()方法，除非组件被配置成使用JITA。

5.当组件使用JITA的时候，决不要调用Dispose()。

评析：

在企业开发中，一种常用的提高性能的方法是采用pooling，特别对于那些很耗资源，构造复杂的对象，就象大家都知道的数据库连接池一样。Enterprise Services允许你在类上通过设置ObjectPoolingAttribute和JustInTimeActivationAttribute来实现对象池。但条件是，ServicedComponent的CanBePooled方法需要返回true，所以如果你的对象想参加对象池的话，那么没有理由不重载CanBePooled来返回true。

一般情形下，一个对象要返回到对象池去，是发生在被garbage collector回收之后。但在一般情形下，garbage collection什么时候回收是不确定的，对此，ServicedComponent实现了一个方法Dispose()可以把对象返回到对象池里去，所以你需要把对象返回到对象池去时，应该调用该ServicedComponent的Dispose方法。

但JustInTimeActivation(JITA，即时激活)则有所不同，JustInTimeActivation是为了实现某些使用模式，由组件自己来控制自己什么时候能被deactivate，而不象普通ObjectPooling对象，是由客户编码来控制的。JustInTimeActivation一般是跟其他服务一起使用的，譬如

在 Enterprise Services 里，设置[Transaction]的话，同时也设置了即时激活服务(还有同步(Synchronization))，所以在方法上设置[AutoComplete]的结果是，当方法返回时，系统会自动释放(deactivate)对象，把对象返回到对象池里去，所以在JITA的情形下调用Dispose是毫无意义的。

6.总是对组件和应用程序设置认证级别。

7.把所有应用程序的认证级别设置成 privacy。

```
[assembly: ApplicationActivation(ActivationOption.Server)]
[assembly: ApplicationAccessControl(true,
    AccessChecksLevel=AccessChecksLevelOption.ApplicationComponent,
     Authentication=AuthenticationOption.Privacy,
    ImpersonationLevel=ImpersonationLevelOption.Identify)]
```

8.把客户端程序集的impersonation level设定成dentity

9.总是把提供服务组件的 ComponentAccessControl 属性设定成 true(这个是默认设置)

```
[ComponentAccessControl]
public class MyComponent : ServicedComponent
{…}
```

10.总是在 Everyone 用户中添加 Marshaler 角色

```
[assembly: SecurityRole("Marshaler", SetEveryoneAccess = true)]
```

11.为所有需要认证的类添加 SecureMethod 属性

```
[SecureMethod]
public class MyComponent : ServicedComponent
{
    …
}
```

评析：

COM+通过declarative方式提供了基于角色的安全服务(RBS)，为enable安全服务，必须设置程序集级的ApplicationAccessControlAttribute为true，否则安全就被disable了。如果要在多个层次上设置RBS，还要把AccessChecksLevel属性设置成AccessChecksLevelOption.ApplicationComponent。而在类级上，则需要设置ComponentAccessControlAttribute(true)，否则系统不会检查该类的方法被调用时的安全权限许可。

一般的安全涉及认证、授权、身份模拟等待的内容。

在 COM+ 里，认证有 6 个级别(详见 5)：

```
RPC_C_AUTHN_LEVEL_NONE
RPC_C_AUTHN_LEVEL_CONNECT
RPC_C_AUTHN_LEVEL_CALL
RPC_C_AUTHN_LEVEL_PKT
RPC_C_AUTHN_LEVEL_PKT_INTEGRITY
RPC_C_AUTHN_LEVEL_PKT_PRIVACY
```

每个层次提供的安全措施如下表所示

http://www.pluralsight.com/keith/images/wiki/guidebookimages/50_1.gif

其中RPC_C_AUTHN_LEVEL_PKT_PRIVACY 提供了最大的安全保证。

在 COM+中，impersonation级别是为了保护客户所设，共有 4 个级别(详见 6)，

```
RPC_C_IMP_LEVEL_ANONYMOUS
RPC_C_IMP_LEVEL_IDENTIFY
RPC_C_IMP_LEVEL_IMPERSONATE
RPC_C_IMP_LEVEL_DELEGATE
```

ANONYMOUS是很怪的级别，如果采用该impersonation级别的话，用户可以通过认证，但服务器端却获取不了用户的security token，也就无法模拟用户的身份。

用IDENTIFY级别的话，服务器端能identify用户，能获取用户的权限情况，但无法模拟用户的身份对资源做正常的操作。

IMPERSONATE 级别允许服务器模拟用户的身份对资源做正常的操作。

DELEGATE级别则允许服务器随便使用用户的network credentials与其他服务器联系。对这些其他服务器而言，这样的连接跟用户直接连接没什么区别。

安全方面的大家 Keith Brown推荐，在远程调用的话，应该使用后面 2 个级别。

如果理解没错的话，Juval Lowy的第 8 个条目(除了Identity 应该是Identify外--可以参考第 7 个条目里的设置)，指的是对客户端程序集的设置，而不是这里所说的服务组件的程序集的设置。那么理解应该是，在服务器回调(callback)时，客户端程序集应该只允许很低的impersonation 级别，这样，服务器就不能对客户端造成“伤害”。

授权方面，你可以使用 SecurityRole Attribute 来在程序集级、类级，甚至方法级上设置基于角色的权限许可。如果设置在程序集级上，在这个角色里的用户可以激活程序集里的任何组件。如果设置在类级上，在这个角色里的用户可以调用组件里的任何方法。对方法级设置权限的话，需要在方法级设置恰当的 SecureMethodAttribute 或 SecurityRoleAttribute。要对方法设置权限控制的话，最容易的方法是在类级上使用 SecureMethodAttribute，然后在方法上设置 SecurityRoleAttribute。

方法级别上设置的 RBS，是通过 DCOM 来实现的。DCOM 会调用一些内部的 interface 的方法，跟这些方法相关的角色名为“Marshaller”。为了保证这些方法的调用成功，一般的做法是把 Everyone 都加入到这个角色里去(详见 3)。

恶意代码的亲密接触

——文件搜索和 API 导址

■ 文 / 温玉洁

在上期介绍了病毒的相关基础知识后，从这期开始我们就要深入病毒内部，开始看一些具体的病毒编码片断，这一期我们会看到API函数搜索以及文件搜索的技术，为后面更加深入的探讨打下良好的基础，如果你已经准备好了，就让作者带你进入这无所不用其极的病毒技术世界吧。

（接上期）

获取 Kernel32.DLL 基址

获取Kernel32.DLL基址的方法很多，最常见的一种是搜索法，如果已知Kernel32.DLL加载的大致地址，那么可由该地址向高地址或低地址进行搜索可以找到其基址。另外一种方法是搜索 NT PEB 结构中的模块列表获取 Kernel32.DLL 的准确加载基址。下面看一下具体的实现代码。

方法 1：暴力搜索获取 Kernel32.DLL 的基址

最初的病毒是指定一个大致的加载地址，比如根据实验在 9X 下其加载地址是 0xBFF70000；在 Windows 2000 下加载基址是 0x77E80000；在 XP 和 2003 下其加载基址是 0x77E60000，因此在NT系统下就可以从0x77e00000开始向高地址搜索，在 9X 下可以从 0xBFF00000 开始向高地址搜索，如果搜索到 Kernel32.DLL 的加载地址，其头部一定是“MZ”标志，由模块起始偏移 0x3C 的双字确定的PE头部标志必然是“PE”标志，因此可根据这两个标志判断是否找到了模块加载地址，也许有人认为该方法不可靠，因为如果恰好有某段数据符合这两个特征，那么找到的基址可能就是错误的，但经实验证明，该判断方法非常可靠，基本不会出现错误。有一点需要注意的是，在所有版本的Windows系统下Kernel32.DLL的加载基址都是按照0x10000对齐的，根据这一特点可以不必逐字节搜索，按照 64K 对齐的边界地址搜索即可。

从大致的一个地址开始搜索Kernel32.DLL基址可能会出现读写到未映射内存区域的情况，因此需要和 SEH 配合使用。如果有在各个版本下准确获取Kernel32.DLL中某地址的通用方法，那么就可以更可靠地从该地址开始向低地址搜索，显然会更加通用。事实上，这种方法是存在的。在系统加载PE文件跳转到PE入口点第一条指令的时候，堆栈顶保存的就是 Kernel32.DLL 中的某个地址，Elkern中采用的就是这种方法：

```
_start:
        pushfd ;If some flags,especial DF,changed,some APIs can crash down!!!
        pushad
_start_@1 equ $
        ;......
        mov ebx,[esp+9*4]
        ;前面已经由 pushfd 和 pushad 压入了 9 个双字
        and ebx,0ffe00000h
        ;该地址为 Kernel32.dll 模块下方的某个地址
        ;先减去 0x100000 确保该地址处于 Kernel32.dll 的下方
        ;向高地址搜索如果将来 Windows 的发行版本中 Kernel32.dll
        ;大小和代码结构发生变化，该方法可能无效
```

ebx 中现在已经是 Kernel32.DLL 基址之前某个地址了，后续代码可以向高地址搜索其基址。该方法有一个缺点，就是必须明确知道程序入口的堆栈指针值，或间接可计算出该值，对于那些在程序入口获取控制权的病毒代码而言，是可以的，但对于采用 EPO 技术的病毒而言，该方法则不适用。事实上还有另外一种更加通用的方法，我们知道在 Win32 程序执行过程中 fs 段寄存器的基址总是指向进程的

TEB，TEB的第一个成员指向SEH链表，该链表每个节点都是一个EXCEPTION_REGISTRATION结构，该结构定义如下：

```
struct EXCEPTION_REGISTRATION{
    struct EXCEPTION_REGISTRATION *prev;
    void* handler;
};
```

在Windows下SEH链表最后一个成员的handler指向Kernel32.DLL中函数UnhandledExceptionFilter的起始地址，利用这一特性我们可以写出更通用的代码：

```
    xor     esi,esi
    lods    dword [fs:esi];取得 SEH 链表的头指针
@@:
    inc     eax         ;是否是最后一个 SEH 节点，检查 prev 是
                        否为 0xFFFFFFFF
    je      @F
    dec     eax
    xchg    esi,eax
    LODSD               ;下一个 SEH 节点
    jmp     near @B
@@:
    LODSD               ;取得Kernel32.dll中UnhandledExceptionFilter的地址
```

在有的病毒直接以0x7FFDE000作为TEB的指针值，其原因在于在Windows 2003 SP1、Windows XP SP2以前的NT类系统上，该值是固定的，这样的确可以节省一两个字节。但是在Windows 2003 SP1、Windows XP SP2中，情况已经发生了变化，出于安全性的考虑，Windows系统开始动态映射TEB了，也就是说，指向TEB的指针值不再固定，因此这种硬编码的方法也就走到了尽头。此时可以按照前面的方法向低地址搜索判断直到找到Kernel32.dll的基址为止。Elkern中判断是否找到了Kernel32.dll基址的相关代码如下：

```
search_api_addr_@1:
        add ebx,10000h
        jz short search_api_addr_seh_restore
        cmp word ptr [ebx],'ZM'         ;是否是 MZ 标志
        jnz short search_api_addr_@1
        mov eax,[ebx+3ch]
        add eax,ebx
        cmp word ptr [eax],'EP'         ;是否具有 PE 标志
        jnz short search_api_addr_@1
       ;找到了 kernel32.dll 的基址
```

方法2：搜索PEB的相关结构获取Kernel32.DLL的基址

前述TEB偏移0x30处，亦即FS:[0x30]地址处保存着一个重要的指针，该指针指向PEB（进程环境块），PEB成员很多，这里并不介绍PEB的详细结构。我们只需要知道PEB结构的偏移0xC处保存着另外一个重要指针ldr，该指针指向PEB_LDR_DATA结构：

```
typedef struct _PEB_LDR_DATA
{

   ULONG       Length;              //  +0x00
   BOOLEAN     Initialized;         //  +0x04
   PVOID       SsHandle;            //  +0x08
   LIST_ENTRY  InLoadOrderModuleList;       //  +0x0c
   LIST_ENTRY  InMemoryOrderModuleList;     //  +0x14
   LIST_ENTRY  InInitializationOrderModuleList; //  +0x1c
} PEB_LDR_DATA,*PPEB_LDR_DATA;          //  +0x24
```

该结构的后三个成员是指向LDR_MODULE链表结构中相应三条双向链表头的指针，分别是按照加载顺序、在内存中的地址顺序和初始化顺序排列的模块信息结构的指针。LDR_MODULE结构如下所示：

```
typedef struct _LDR_MODULE
{
    LIST_ENTRY          InLoadOrderModuleList;      //  +0x00
    LIST_ENTRY          InMemoryOrderModuleList;    //  +0x08
    LIST_ENTRY          InInitializationOrderModuleList; //  +0x10
    PVOID               BaseAddress;                //  +0x18
    PVOID               EntryPoint;                 //  +0x1c
    ULONG               SizeOfImage;                //  +0x20
    UNICODE_STRING      FullDllName;                //  +0x24
    UNICODE_STRING      BaseDllName;                //  +0x2c
    ULONG               Flags;                      //  +0x34
    SHORT               LoadCount;                  //  +0x38
    SHORT               TlsIndex;                   //  +0x3a
    LIST_ENTRY          HashTableEntry;             //  +0x3c
    ULONG               TimeDateStamp;              //  +0x44
                                                    //  +0x48
} LDR_MODULE, *PLDR_MODULE;
```

Peb->Ldr->InInitializationOrderModuleList指向按照初始化顺序排序的第一个LDR_MODULE节点的InInitializationOrderModuleList成员的指针，在WinNT平台（不包含Win9X）下，该链表头节点的LDR_MODULE结构包含的是NTDLL.DLL的相关信息，而链表的下一个节点所包含的就是Kernel32.dll相关的信息了，该节点LDR_MODULE结构中的BaseAddress不正是我们所苦苦寻找的吗。注意InInitializationOrderModuleList是LDR_MODULE的第3个成员，因此要获取BaseAddress的地址，只需将其指针加8再derefrence即可。因此下面的汇编代码即可获取Kernel32.DLL的基址：

```
mov eax, dword ptr fs:[30h]
;获取 PEB 基址
mov eax, dword ptr [eax+0ch]
;获取 PEB_LDR_DATA 结构指针
mov esi, dword ptr [eax+1ch]
;获取 InInitializationOrderModuleList 链表头第一个 LDR_MODULE 节点
InInitializationOrderModuleList 成员的指针
```

```
lodsd
;获取双向链表当前节点后继的指针
mov ebx, dword ptr [eax+08h]
;取其基地址，该结构当前包含的是

;kernel32.dll 相关的信息
```

该方法在所有的Windows NT（包括Windows 2003 SP1和Windows XP SP2）操作系统上都是有效的，唯一的缺憾是由于PEB结构不同，该方法在Win9X系统上无效。听起来可能比较费解，还是用一张图更加清晰一些（见图6）。

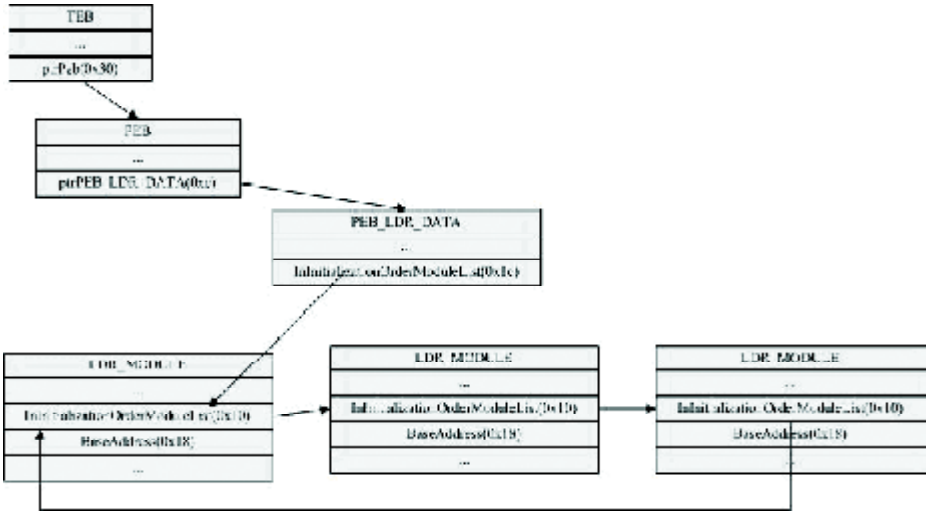


图6：利用PEB搜索kernel32.dll基地址的过程

解析PE文件的导出函数表

PE文件的函数导出机制是进行模块间动态调用的重要机制，对于正常的程序，相关操作是由系统加载器在程序加载前自动完成的，对用户程序是透明的。但要想在病毒代码中实现函数地址的动态解析以取代加载器，那就有必要了解函数导出表的结构了。在图1中可以看到在PE头结构IMAGE_OPTIONAL_HEADER32结构中包含一个DataDirectory数组结构，该结构包含16个成员，每个成员都是一个IMAGE_DATA_DIRECTORY结构：

```
typedef struct _IMAGE_DATA_DIRECTORY {
    DWORD   VirtualAddress;
    DWORD   Size;
} IMAGE_DATA_DIRECTORY, *PIMAGE_DATA_DIRECTORY;
```

DataDirectory数组的每个结构都指向一个重要的数据结构，第一个成员指向导出函数表（索引0），第2个成员指向PE文件的引入函数表（索引1）。DataDirectory中的第一个成员指向导出函数表的IMAGE_EXPORT_DIRECTORY结构：

```
typedef struct _IMAGE_EXPORT_DIRECTORY {
    DWORD   Characteristics;
    DWORD   TimeDateStamp;
    WORD    MajorVersion;
    WORD    MinorVersion;
    DWORD   Name;
    DWORD   Base;
    DWORD   NumberOfFunctions;
    DWORD   NumberOfNames;
    DWORD   AddressOfFunctions;     // RVA from base of image
    DWORD   AddressOfNames;         // RVA from base of image
    DWORD   AddressOfNameOrdinals;  // RVA from base of image
} IMAGE_EXPORT_DIRECTORY, *PIMAGE_EXPORT_DIRECTORY;
```

AddressOfFunctions是一个双字数组，包含了所有导出函数的RVA，另外两个成员AddressOfNames也是一个双字数组，包含了指向导出函数名字的字符串的RVA，AddressOfNameOrdinals是一个字数组（16bit），和AddressOfNames数组是并行的，和AddressOfNames数组一起确定了相应引出函数的序号，该序号可直接用于索引AddressOfFunctions数组获取导出函数的地址。因此病毒搜索指定的API就包含了如下步骤：

a)获取NumberOfNames的值以及AddressOfNames、AddressOfNameOrdinals和AddressOfFunctions的数组的地址。

b)搜索AddressOfNames数组，按字符串对比，若找到相应的API，转d

c)若NumberOfNames名字尚未全部搜索完毕，转b继续搜索，若搜索完毕，则表明未找到进行错误处理，这一步通常可以省略，因为我们已经知道相应的DLL中肯定导出了相应的函数。

d）获取当前函数名字指针在AddressOfNames数组中的索引，在AddressOfNameOrdinals数组中取出以该值索引的函数序号，以该序号值作为AddressOfFunctions数组的索引，在AddressOfFunctions数组中取出导出函数的RVA值，加上基址就得到了运行时导出函数的地址。

看起来似乎比较罗嗦，实际上这是PE设计时为考虑灵活性而做出的牺牲。不过实现起来还是比较简单的，通常汇编代码编译后不到100字节。以下是在Kernel32搜索GetProcAddress的完整代码：

```
        push    esi
        ;esi=VA Kernel32.BASE
        ;edi=RVA K32.pehdr
        mov     ebp,esi
        mov     edi,[ebp+edi+peh.DataDirectory]

        push    edi esi
```

```
        mov     eax,[ebp+edi+peexc.AddressOfNames]
        mov     edx,[ebp+edi+peexc.AddressOfNameOrdinals]
        call    @F
        db      "GetProcAddress",0
@@:
        pop     edi
        mov     ecx,15
        sub     eax,4
next_:
        add     eax,4
        add     edi,ecx
        sub     edi,15
        mov     esi,[ebp+eax]
        add     esi,ebp
        mov     ecx,15
        repz    cmpsb
        ;进行字符串比较，判断是否为要查找的函数
        jnz     next_

        pop     esi edi

        sub     eax,[ebp+edi+peexc.AddressOfNames]
        shr     eax,1
        add     edx,ebp
        movzx   eax,word [edx+eax]
        add     esi,[ebp+edi+peexc.AddressOfFunctions]
        add     ebp,[esi+eax*4]
        ;ebp=Kernel32.GetProcAddress.addr
                ;use GetProcAddress and hModule to get other func
        pop     esi                     ;esi=kernel32 Base
```

在前面解析导出函数表获取API地址的时候，采用的是直接比较字符串的方法判断是不是找到了相应的API，其实还可以计算函数名字的hash，然后同预计算的hash进行比对，现代的PE病毒更多采用的hash的方法，其原因在于一般的函数名字长度都大于4字节，而用hash只要占用4个字节或2个字节，可以节省空间，此外还有抗病毒分析的作用，因为hash要比字符串名字费解得多。hash算法的设计只要能保证无冲突即可，可以用crc等成熟算法，也可以设计自己的简单算法。在Elkern中就使用了crc16算法。

文件搜索

文件搜索是病毒的重要功能模块之一，也是实现感染和传播的关键。现代Windows和各种移动介质的文件系统可能采用多种复杂格式，因此象一些Dos病毒一样试图直接存取文件系统（读写扇区）是不大现实的。通常利用Win32 API的FindFirstFile和FindNextFile来实当前目录下所有目录和文件的搜索，通过判断搜索到的文件属性，可区分是否为目录或可执行文件，对于可执行文件则根据预先设计好的感染策略进行感染；对于当前目录下的所有子目录以及特殊的..父目录，可以使用递归或非递归的方式利用上述两个API全部进行遍历，因此从某个驱动器或网络共享文件夹的任意一个子目录开始，都可以遍历当前驱动器或网络共享文件夹内的所有文件和目录。一般地，搜索文件从驱动器或共享文件夹的根目录开始，那么如何得到当前系统中存在的所有驱动器或所有的共享文件夹列表呢？对于前一个问题，我们知道Windows下可划分A:～Z:共26个逻辑盘符，因此可以从A:开始递增搜索所有的驱动器，使用Win32 API GetDriveType判断当前搜索的盘符是否存在，以及是否是固定硬盘、可移动存储介质、是否可写或是网络驱动器等。一般病毒只感染固定硬盘或网络驱动器。由于汇编语言在表述算法时显得过于冗长，因此算法部分使用C语言描述，当然将C算法转换成汇编语言是很简单的过程。

下面的代码enumdisk.cpp将显示A-Z各个驱动器的相关属性：

```
#include <windows.h>
#include <stdio.h>

#define MAX_DRIVENAME_LENGTH    64
void __cdecl main(int argc,char *argv[])
{
    char DriveName[MAX_DRIVENAME_LENGTH];
    char *p;
    unsigned int drv_attr;

    p = DriveName;
    strncpy(DriveName,"A:",MAX_DRIVENAME_LENGTH);

    for(;*p<'Z';++*p) {
        drv_attr = GetDriveType(p);

        switch(drv_attr)
        {
        case DRIVE_UNKNOWN:             // 未知类型
            printf("drive %s type %s\n",p,"DRIVE_UNKNOWN");
            break;
        case DRIVE_NO_ROOT_DIR:         // 该驱动器不存在
            printf("drive %s type %s\n",p,
                "DRIVE_NO_ROOT_DIR");break;
        case DRIVE_REMOVABLE:  // 可移动盘，软盘或U盘或
                               // 移动硬盘等
            printf("drive %s type %s\n",p,"DRIVE_REMOVABLE");
            break;
        case DRIVE_FIXED:           // 固定硬盘
            printf("drive %s type %s\n",p,"DRIVE_FIXED");break;
        case DRIVE_REMOTE:      // 一般是映射网络驱动器
            printf("drive %s type %s\n",p,"DRIVE_REMOTE");break;
        case DRIVE_CDROM:       // 光盘
            printf("drive %s type %s\n",p,"DRIVE_CDROM");break;
        case DRIVE_RAMDISK:     // RAM DISK
            printf("drive %s type %s\n",p,"DRIVE_RAMDISK");
            break;
        }
    }
}
```

与仅仅显示一条信息不同的是，病毒此时将调用文件枚举函数（如后面给出的enum_path函数）从当前根目录开

始遍历DRIVE_FIXED的驱动器上的所有文件，根据预定义策略进行文件感染。

网络共享资源也是按树状组织的，非叶节点称为容器（container），对容器需要进一步搜索直到到达叶子节点为止，叶子节点才是共享资源的根路径。共享资源一般分成两种：共享打印设备和共享文件夹。对于网络共享文件的搜索，采用WNetOpenEnum和WNetEnumResource（由mpr.dll导出）进行递归枚举。其函数原型及参数含义请参阅MSDN，使用如下代码enumshare.cpp将显示所有的网络驱动器共享文件夹的路径：

```
#include <windows.h>
#include <stdio.h>
#pragma comment(lib,"mpr.lib")

int enum_netshare(LPNETRESOURCE lpnr);

void __cdecl main(int argc,char *argv[])
{
    enum_netshare(0);
}


int enum_netshare(LPNETRESOURCE lpnr)
{
    DWORD r, rEnum,usage;
    HANDLE hEnum;
    DWORD cbBuffer = 16384;
    DWORD cEntries = -1;
    LPNETRESOURCE lpnrLocal;
    // NETRESOURCE数组结构的指针
    DWORD i;


    r = WNetOpenEnum(RESOURCE_GLOBALNET,
      // 范围:所有网络资源
                     RESOURCETYPE_DISK,
     // 类型:仅枚举可存储介质
                     RESOURCEUSAGE_ALL,
     // 使用状态:所有
                     lpnr,
    // 初次调用时为 NULL
                     &hEnum);
    // 成功后返回的网络资源句柄

    if (r != NO_ERROR) {
        printf("WNetOpenEnum error....\n");
        return FALSE;
    }

    lpnrLocal = (LPNETRESOURCE) malloc(cbBuffer);
    if (lpnrLocal == NULL)
        return FALSE;

    do
    {
        ZeroMemory(lpnrLocal, cbBuffer);

        rEnum = WNetEnumResource(hEnum,
                    &cEntries, // 返回尽可能多的结果
                      lpnrLocal,      // LPNETRESOURCE
                                &cbBuffer);    // buffer大小
        if (rEnum == NO_ERROR) {

            for(i = 0; i < cEntries; i++) {

                usage = lpnrLocal[i].dwUsage;

                if(usage & RESOURCEUSAGE_CONTAINER) {

                    if(!enum_netshare(&lpnrLocal[i]))
                        printf("Errors detected in enum
                                process...\n");
                }else{

                    // 这里病毒可调用遍历函数遍历该
                    // 共享文件夹下的所有文件
                    // enum_path(lpnrLocal[i].lpRemoteName);
                     printf("find %s --> %s\n",
                      lpnrLocal[i].lpLocalName,
                      lpnrLocal[i].lpRemoteName);
                }
            }
        }else if (rEnum != ERROR_NO_MORE_ITEMS) {
            printf("WNetEnumResource error...\n");
            break;
        }
    }while(rEnum != ERROR_NO_MORE_ITEMS);

    free((void*)lpnrLocal);

    r = WNetCloseEnum(hEnum);

    if(r != NO_ERROR) {
        printf("WNetCloseEnum error....\n");
        return FALSE;
    }

    return TRUE;
}
```

遍历开始时WNetOpenEnum第4形参为0，在发现共享容器进行递归调用时候，该参数将为共享容器的NETRESOURCE结构指针。从NETRESOURCE结构中可以找到我们感兴趣的lpRemoteName，该指针不为0则表示是有效的共享容器或共享文件夹。

```
typedef struct _NETRESOURCE {
  DWORD dwScope;
  DWORD dwType;
  DWORD dwDisplayType;
  DWORD dwUsage;
  LPTSTR lpLocalName;
  LPTSTR lpRemoteName;
  LPTSTR lpComment;
  LPTSTR lpProvider;
} NETRESOURCE;
```

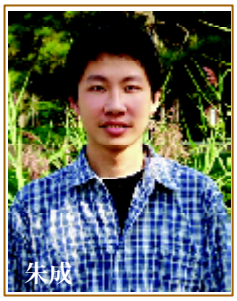
（未完待续）

■ 责任编辑：崔轶（cuiyi@csdn.net）

杀毒软件技术的亲密接触

——静态查毒引擎的实现





■ 文 / 马杰　朱成

病毒与反病毒技术在很多朋友眼中总有一些神秘感。在接下来的几期文章中，我们会邀请Rising的技术专家向大家介绍关于杀毒软件所用到的方方面面的技术，以求使大家更好的明白杀毒软件到底是如何工作的，同时使大家了解到杀毒技术是如何随着病毒技术的日益变化逐步发展到今天的。在这个过程中，我们会写一系列代码来演示相关的技术，虽然简单，但那可是一个真实引擎的原型。

病毒大致分为以下几类：传统病毒，宏病毒，恶意脚本，木马、黑客、蠕虫、破坏性程序。

病毒与引擎的变迁

简单特征码

80年代末期，基于个人电脑病毒的诞生，随即就有了清除病毒的工具——反病毒软件。这一时期，病毒所使用的技术还比较简单，从而检测相对容易，最广泛使用的就是特征码匹配的方法。

特征码是什么呢？比如说，“如果在第1034字节处是下面的内容:0xec ，0x99，0x80，0x99，就表示是大麻病毒。”这就是特征码，一串表明病毒自身特征的十六进制的字串。特征码一般都选得很长，有时可达数十字节，一般也会选取多个，以保证正确判断。杀毒软件通过利用特征串，可以非常容易的查出病毒。

广谱特征

为了躲避杀毒软件的查杀，电脑病毒开始进化。病毒为了躲避杀毒软件的查杀，逐渐演变为变形的形式，每感染一次，就对自身变一次形，通过对自身的变形来躲避查杀。这样一来，同一种病毒的变种病毒大量增加，甚至可以到达天文数字的量级。大量的变形病毒不同形态之间甚至可以做到没有超过三个连续字节是相同的。

为了对付这种情况，首先特征码的获取不可能再是简单的取出一段代码来，而是分段的，中间可以包含任意的内容（也就是增加了一些不参加比较的“掩码字节”，在出

1、传统病毒：能够感染的程序。通过改变文件或者其他东西进行传播，通常有感染可执行文件的文件型病毒和感染引导扇区的引导型病毒；

2、宏病毒（Macro）：利用Word、Excel等的宏脚本功能进行传播的病毒；

3、恶意脚本（Script）：做破坏的脚本程序。包括HTML脚本、批处理脚本、VB、JS脚本等；

4、木马（Trojan）程序：当病毒程序被激活或启动后用户无法终止其运行。广义上说，所有的网络服务程序都是木马，判定是否是木马病毒的标准不好确定，通常的标准是：在用户不知情的情况下安装，隐藏在后台，服务器端一般没有界面无法配置；

5、黑客(Hack) 程序：利用网络来攻击其他计算机的网络工具，被运行或激活后就象其他正常程序一样有界面;黑客程序是用来攻击/破坏别人的计算机，对使用者本身的机器没有损害；

6、蠕虫（Worm）程序：蠕虫病毒是一种可以利用操作系统的漏洞、电子邮件、P2P软件等自动传播自身的病毒；

7、破坏性程序（Harm）：病毒启动后，破坏用户计算机系统，如删除文件，格式化硬盘等。常见的是bat文件，也有一些是可执行文件，有一部分和恶意网页结合使用。

现“掩码字节”的地方，出现什么内容都不参加比较)。这就是曾经提出的广谱特征码的概念。这个技术在一段时间内，对于处理某些变形的病毒提供了一种方法，但是也使误报率大大增加，所以采用广谱特征码的技术目前已不能有效的对新病毒进行查杀，并且还可能把正规程序当作病毒误报给用户。

特征码技术	静态广谱特征扫描技术	启发式扫描技术（静态扫描＋未知特征）	行为判定技术
速度快，准确率高	可以检测部分变形病毒	误报率低，能检测变形病毒和病毒变种	基于强大而完整的虚拟机技术能够对未知病毒进行判别，对标准病毒准确率高
不能对付变形病毒或加密病毒	误报率高	对未知病毒的检测能力较低	实现难度大，速度慢

启发式扫描

为了对付病毒的不断变化和对未知病毒的研究，启发式扫描方式出现了。启发式扫描是通过分析指令出现的顺序，或特定组合情况等常见病毒的标准特征来决定文件是否感染未知病毒。因为病毒要达到感染和破坏的目的，通常的行为都会有一定的特征，例如非常规读写文件，终结自身，非常规切入零环等等。所以可以根据扫描特定的行为或多种行为的组合来判断一个程序是否是病毒。

行为判定

针对变形病毒、未知病毒等复杂的病毒情况，极少数杀毒软件采用了虚拟机技术，达到了对未知病毒良好的查杀效果。它实际上是一种可控的，由软件模拟出来的程序虚拟运行环境，就像我们看的电影《黑客帝国》一样。在这一环境中虚拟执行的程序，就像生活在母体(Matrix)中的人，不论好坏，其一切行为都是受到建筑师(architect)控制的。虽然病毒通过各种方式来躲避杀毒软件，但是当它运行在虚拟机中时，它并不知道自己的一切行为都在被虚拟机所监控，所以当它在虚拟机中脱去伪装进行传染时，就会被虚拟机所发现，如此一来，利用虚拟机技术就可以发现大部分的变形病毒和大量的未知病毒。

引擎技术对比

各种引擎技术相比，虚拟机就像是一个侦探，可以根据对人的行为识别犯罪活动；启发式扫描就像是警察，看你身上携带了枪支而怀疑你；广谱特征是拿着照片追查已知的罪犯，但是会注意是否带了假发或者墨镜来逃避检查；而特征码识别就只是通过对人的外貌来判断。

简单总结一下各种引擎技术的优缺点：

解压缩与去壳

病毒隐藏自身的方法还有加壳和压缩两种方法。加壳是通过一系列的数学运算，将可执行程序或动态链接文件的编码进行改变，以达到缩小程序体积或加密程序编码的目的。通常常见的加壳工具有UPX、ASPack等。病毒通过使用不同种类或者版本的加壳软件，对自身进行加壳，使得杀毒软件无法发现真正的病毒体，以逃避查杀。并且由于加壳的工具种类很多，同一个工具也存在不同的版本。为了检测已加壳的病毒，就必须要针对不同种类不同版本的壳编写脱壳程序才可以发现壳内隐藏的真正病毒体。所以，杀毒软件对病毒的查杀能力也在一定程度上取决于他自身的脱壳能力。

压缩是普通用户日常经常使用的减小文件体积的方法，常见的工具软件有WinZip、WinRAR等。病毒有时候会隐藏在压缩包内部，如果一个杀毒软件没有解压缩的能力就不可能查杀压缩包内的病毒。同时，如果杀毒软件不具备相应格式的压缩能力，在查杀病毒后就不能复原压缩包，导致压缩包破坏。可见，杀毒软件要想做到对病毒的全面捕获与查杀，脱壳解压能力也是至关重要的。

利用特征码技术的静态杀毒引擎

特征码的选取

在进入程序的详细讲解之前，先来讲一下对于病毒程序的特征码通常是如何选取出来的，以及特征码的结构是什么样子的。

通常选择特征码是按照以下思路。

1、获取一个病毒程序的长度，根据长度可以将文件分为几份，份数根据样本长度而定，可以是3~5份，也可以更多。分成几段获取特征码的方法可以很大程度上避免采用单一特征码误报病毒现象的发生，也可以避免特征码过于集中造成的误报。

2、每份中选取通常为16或32个字节长的特征串。

在选取时，应该采取如下的原则：

1、如果选出来的信息是通用信息，即很多文件该位置

都是一样的信息，那么舍弃，调整偏移量后重新选取。

2、如果选取出来的信息是全零的字节。那么也要调整偏移后重新选取。

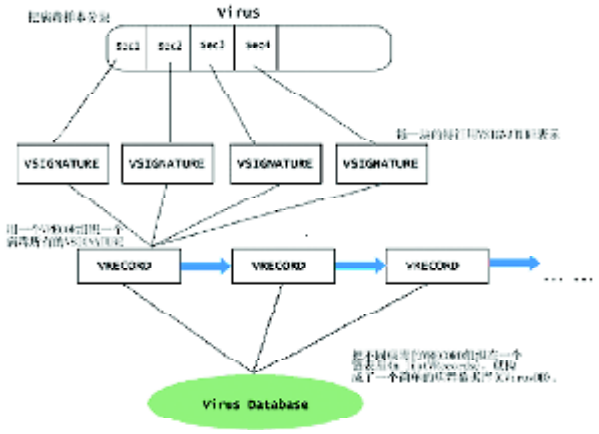
关键数据结构

这一部分我们来解决病毒库的问题，顾名思义，这是一个记录病毒二进制特征的数据库。在本文这个例子中，我们使用如下的方法来记录每一个病毒：

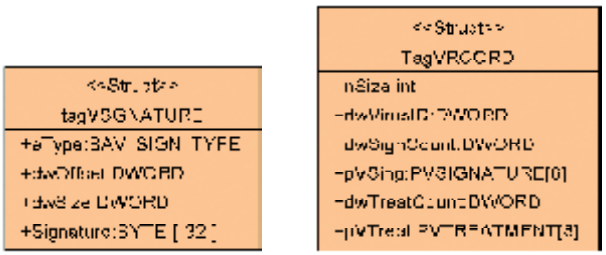
1、我们把一个病毒分成不同的块，在每一块中取一个特征，用一个名为VSIGNATURE的结构（下面会详述）来描述；

2、用一个名为VRECORD的结构来把一个病毒的所有VSIGNATURE组织在一起，这样就构成了对一个病毒的完整描述；

3、把这些完整的描述用链表组织起来，就是我们的病毒数据库了。如图所示：



下面就来看看上面提到的两个重要结构VSIGNATURE和VRECORD。一个VSIGNATURE是一个特征，很多个特征组成了一条病毒记录，也就是一个VRECORD。




```
typedef struct tagVSIGNATURE
{
        BAV_SIGN_TYPE    eType;
        DWORD                   dwOffset;
        DWORD                   dwSize;
        BYTE                    Signature[MAX_SIGNATURE_LEN];
}VSIGNATURE,*PVSIGNATURE;
```

```
typedef struct tagVRECORD
{
        int             nSize;
        DWORD           dwVirusID;
        DWORD           dwSignCount;
        PVSIGNATURE     pVSing[8];
        DWORD           dwTreatCount;
        PVTREATMENT     pVTreat[8];
}VRECORD,*PVRECORD;
```

VSIGNATURE结构是用于存放单一特征码的，其中的eType成员变量是一个枚举结构，用来定义特征码的类型，这里演示工程里目前我们只定义了一种简单文件特征。dwOffset成员存储该特征码的偏移量。dwSize成员存储特征码的长度。字节型的Signature数组成员存放特征码串，最大长度由MAX_SIGNATURE_LEN宏控制，目前为32字节。

VRECORD结构用于存放病毒库中每个病毒记录的内容。其中nSize用于控制结构的版本，目前我们不用过多关心。dwVirusID成员指定病毒的ID编号。dwSignCount成员存放特征码（VSIGNATURE）的段数，也就是对于该病毒取了多少段特征码。pVSing数组成员存放每段特征码的内容，这里用数组是为了演示方便，以后我们会改为可变长度的数据结构。dwTreatCount成员存放处理该病毒的方法数量。pVTreat数组成员存放处理该病毒的每种方法的内容。这两个成员要到我们增加杀毒方法的时候才会用到。

下面来看一个具体的例子，eicar测试病毒的第一条特征是这样的：

```
{
        BS_PHY_FILE, 0, 32,
        0x58, 0x35, 0x4F, 0x21, 0x50, 0x25, 0x40, 0x41,
        0x50, 0x5B, 0x34, 0x5C, 0x50, 0x5A, 0x58, 0x35,
        0x34, 0x28, 0x50, 0x5E, 0x29, 0x37, 0x43, 0x43,
        0x29, 0x37, 0x7D, 0x24, 0x45, 0x49, 0x43, 0x41,
}
```

它表明这是一个简单文件特征，特征起始地址0，特征长度32，后面32个字节是具体特征值。

库与引擎类

清楚了病毒库的构成后，接下来我们看一下具体的扫描方法。关键要把引擎、库与被扫描对象之间的基本关系和分工搞清楚。

引擎（CEngine）负责被扫描对象的遍历，对应到目前版本的代码上，引擎遍历目录，将找到的文件生成被扫描对象（CScanObj）交给当前病毒库对象的Search()方法。而病毒数据库对象（CVirusDB）则负责在自己管理的库中搜

索。为了演示程序的简单，这一版的病毒库没有从文件中加载，而是直接在CVirusDB::Load()中编码进去的。从整体上来看，搜索过程经历了下面3个阶段：

1．CEngine遍历目录，生成被扫描对象并把该对象传递到病毒数据库中m_pcVDB->Search(&cScanObj)

2．病毒数据库搜索自身的记录CVirusDB::Search()，并调用被扫描对象CFileObject的Compare()方法进行匹配；

3．被扫描对象返回匹配结果，病毒库对象(CVirusDB)根据结果来决定是否是病毒，并返回病毒ID。

下面就来仔细看看以上提到的每一个关键部分：

首先是引擎的目录遍历。CEngine类中DFS()函数是负责文件系统深度优先搜索的函数，其中以下一段就是产生一个对象，然后传递给CVirusDB::Search()方法来查毒：

```
{
        m_cScanResults.dwObjCount++;

        CFileObject          cScanObj;
        cScanObj.m_eObjType          = BO_PHY_FILE;
        cScanObj.m_strObjName        = lpszPathName;
        if( !cScanObj.Open() )
        {
                // TODO: show error here.
                return;
        }

        DWORD dwVID = m_pcVDB->Search(&cScanObj);
        if( dwVID )
        {
                PSCAN_RECORD  pScanRecord = new SCAN_RECORD;
                if(pScanRecord)
                {
                        CFileObject* pScanObj = new
                                        CFileObject(cScanObj);

                        pScanRecord->dwVirusID      = dwVID;
                        pScanRecord->eResult        =
                                                BR_WITH_VIRUS;
                        pScanRecord->pScanObject=  pScanObj;
                        pScanRecord->pNext =
                                m_cScanResults.pScanRecords;
                        m_cScanResults.pScanRecords = pScanRecord;
                        m_cScanResults.dwRecCount++;
                }
        }

        cScanObj.Close();
}
```

其次是病毒数据库的搜索工作。CVirusDB类中的Search函数是用于在病毒库中匹配特征的成员函数，内容如下：

```
DWORD CVirusDB::Search(CScanObject* pScanObj)
{
        list<PVRECORD>::iterator    iter = m_listVRecords.begin();

        while(iter!=m_listVRecords.end())
        {
                PVRECORD        pVRec = *iter;
                ASSERT(pVRec);
                if(pVRec)
                {
                        bool bVirus = true;
                        for(unsigned int i=0; i<pVRec->dwSignCount;
                           i++)
                        {
                                bVirus &= pScanObj->Compare
                                   (pVRec->pVSing[i]->dwOffset,
                                   pVRec->pVSing[i]->dwSize,
                                    pVRec->pVSing[i]->Signature);
                                if(!bVirus)     break;
                        }

                        // match all signatures
                        if(bVirus)
                        {
                                return pVRec->dwVirusID;
                        }
                }
                else
                {
                        // error
                        return 0xFFFFFFFF;
                }

                iter++;
        }

        // no match record in VirusDB
        return 0;
}
```

While循环是遍历本病毒库中所有的记录。在For循环中，根据指向VRECORD结构的指针pVRec中dwSignCount的内容可知特征码的数量。然后循环用pScanObj->Compare()函数比较每段特征码与文件中指定偏移处的内容是否一致，如果全部一致，则说明该文件是病毒。

最后，被扫描对象进行特征串的匹配其实就是memcmp，这一步没有什么特别的。

本次主要为大家介绍了根据特征码查病毒的方法，源代码可以在网站http://bav.netsv.org下载。在真实的环境中，查毒引擎的设计要比这个例子复杂上很多，一般都有多个库，还利用了诸如脱壳，解压，虚拟机等等技术，这些我们将在日后的文章中逐渐介绍。

（读者可以到杂志频道下载完整代码）

作者介绍：马杰，北京瑞星科技股份有限公司研发部副总经理。从事反病毒、反垃圾和网络安全工作多年，设计开发了瑞星杀毒软件、企业级防火墙、企业级防毒墙等多个产品。

朱成，北京瑞星科技股份有限公司研发部网络攻防实验室研究员。从事网络安全及反病毒研究多年，精通网络安全设备的原理与部署，拥有多年的系统渗透测试经验。

■ 责任编辑：崔轶（cuiyi@csdn.net）

基于模式的UI设计实战


卫昆

■ 文 / 卫昆　彭福祥

在上一期文章中作者留下了一个悬念：什么样的GoF模式适合架构UI框架中的三种模型呢？作者通过对环境操控模型、导航逻辑模型等方面给出了答案。

环境操控模型是一种主动模型，用它来控制导航逻辑模型和业务表示模型。它分为浏览、增加、修改、删除、取消、保存等数据状态，开发人员也可以动态地增加新的状态以满足新的开发需求。由于状态众多，选择哪一种状态完全由当前发生的业务决定。在期间的每个操作状态负责对一个操作对象在其生命周期中满足某种条件的一个时间段建模，当一个业务发生时，客户端会触发操作状态间的转换机制，从而导致对象从一种操作状态转化到另一种新的操作状态。

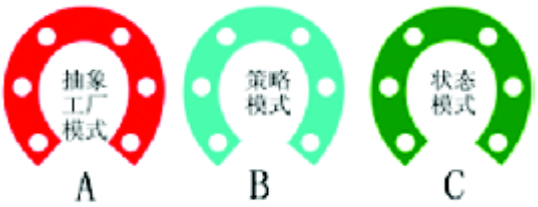


图1：环境操控模型被选方案

我们知道，A模式属创建型模式的范畴，B模式和C模式同属于行为类模式的范畴。那么在GOF模式的整体分类中，分类的标准是怎样制定的？因为不搞清楚它们的分类标准，不可能选择出将要架构的正确模式。

创建型模式：抽象了实例化过程，能帮助一个系统独立于如何创建、组合和表示它的对象范围。创建型模式分为类创建型模式和对象创建型模式。所谓类创建型模式是指使用了继承关系，把类的构建延迟到派生类中。而对象创建型模式是把对象的创建过程委派给另一个对象，从而动态地决定客户端将得到哪些具体类的实例。

结构型模式：是一组考虑到类如何构建及对象如何组合成大型结构的模式。包括类对象结构模式和对象结构模式。类对象结构模式是使用继承机制来组合接口实现。例如通过多重继承的机制组合两个或两个以上的类成为一个类或接口。对象结构模式是对象如何组合以产生新功能的模式。

行为型模式：是对象间的算法（algorithms）及职责的划分。它们将设计者的注意力从流程控制转移到对象的互动方式。考虑的不只是对象及类别的样式同时也是其对象的交互与沟通。包括类行为型模式和对象行为模式。其中类行为模式是使用类的继承机制在其间分配行为。而对象行为模式是采用对象的合成行为来分配行为。

因抽象工厂模式属创建型模式的范畴，具有该分类的典型特征。该模式主要应用在创建对象的接口，并让子类决定将哪一个类该实例化。也就是说，当你编译时不知道要实例化哪一个类，这时侯就可以把该类的创建放在子类中，并提供专门的接口创建一个对象。因此，抽象工厂模式专用于以下几点：

1、使用一个类库，只想显示它们的接口，不用显示它们具体的实现，从而使使用者不依赖于具体的实现。

2、客户端与它们的产品类如何被创建、组合、应用及表现无关。

3、当要强调一系列产品的相关对象的设计以便进行联合使用时。

4、一个系统要由多个产品系列中的一个接口来配置，应选择此模型。

这些特征不符合我们将要设计的环境操控模型，环境操控模型主要体现的宗旨是通过数据状态的产生和传递来驱动依附于它的模型，而不是强调一个创建过程，是强调一个数据的流动、通讯的过程，是一种主控对象与被控对象的通讯机制。

至此，我们抛弃了A答案，答案只能在B和C中产生。

看过GOF的《设计模式》书的人可能都清楚，同属于行为型模式分类的策略模式和状态模式的确难以确分，在某些情况下他们就像一对孪生兄弟，有时它们却有着天壤之别的差距。但是具体应用哪一种模式还是有规可循的。

首先说说策略模式。策略模式（Strategy）是针对一组算法，并将每一个算法都封装到具有共同接口的独立的类中，使得算法之间可以相互替换，从而使得算法可以在不影响到客户端的情况下发生变化。它把系统的行为和环境巧妙地分开。环境类负责维持和查询行为类，而各种算法可在具体的策略类中提供。由于把算法和环境的分开，算法的管理都不会间接地影响到环境和客户端。

举个形象的例子来说明这一点，譬如我们在商场购物，手中的现金是有限的，可是琳琅满目的商品却令我们眼花缭乱，只能根据自己的实际经济状况购买合适的商品。衣服只能买300元钱之内的，体育用品也只能买一些中档货了。如果衣服买200元之内的，体育用品的档次也可以在上升一些。这个想法也不错。在如果，不买衣服，只买体育用品？……没办法，我们只能用有限的钱，揣摸灵活的消费方案去购买一些商品。

再例如我们装一台价格在6000元以内的计算机。这需要我们精心考虑所选用器件的价格、性能与质量。假如我们从事绘图行业（如：三维制作），显卡就得买高档一些的，显示器最好在纯平19寸以上的，而别的器件的价位可以稍稍降低一些。如果我们的计算机应用不大，只是作为一种家庭摆设，我们可以选择外形比较漂亮的机型。因此，通过不同的个人需要来选择不同种类的计算机的想法就是一个典型的策略模型。所以，策略模式是一种被选择的模式，是一种被动的方法和算法集合，它们是被选择的方案，是被动的对象。

主要适用于：

1、当许多相关的类只是行为有异时，在一组稍显差异的行为中能够选择一种行为来表示时。

2、当实现一个行为有多种算法时，也可以考虑应用策略模式。这些行为被封装到派生类中，拥有共同的接口或继承于一个抽象类。客户端只需要选择抽象类的对象就可以利用此抽象类下不同的派生类的行为。

3、当一组算法类不需要暴露给客户时。

4、当想减弱系统中出现的if……..else语句时，因为这些语句往往扮演着不同方案，可以考虑应用策略模式。

再说说状态模式。状态模式（State）是一种允许一个对象在其内部状态改变的时候而改变行为的模式。这个对象看上去像是改变了它的类一样。状态模式把所研究的对象的行为包装在不同的状态对象里，每一个状态对象都属于一个抽象状态类的一个子类。状态模式的意图是让一个对象在其内部状态改变的时候，其行为也随之改变。状态模式需要对每一个系统可能取得的状态创立一个状态类的子类。当系统的状态变化时，系统便改变所选的子类。它是一种用主动态度来改变自己从而达成不同的协议或形成不同的效果的一种模式，强调的是一种主观行为，是一种行为的缔造者。

举个例子更能彻底地说明这一点。现实生活里水物质的各种状态的变化现象就很贴切的印证了状态模式的实质。水在温度高于100度的时候可以转变为气态，低于0度时转变为固态，而在常温下则是液态。因此，在这种现象中温度是一个改变对象的外部条件，而气态、固态、液态就是水物质的不同状态。

因此该模式主要适用于：

1、一个对象的行为取决于它的状态，并且它必须在运行时刻根据状态改变它的行为。

2、一个操作中含有庞大的多分支的条件语句，且这些分支依赖于该对象的状态。这个状态通常用一个或多个枚举常量表示。通常，有多个操作包含这一相同的条件结构。State模式将每一个条件分支放入一个独立的类中。这使得你可以根据对象自身的情况将对象的状态作为一个对象，这一对象可以不依赖于其他对象而独立变化。

表1总结了以上的两种模式的区别，请读者参考。

表1

GOF模式	所处的位置	编程角度
策略模式	被动区	方案的集合
状态模式	主动区	自身变化归集的方案集合

看了表1，读者应该清楚我们这道题目的正确答案了，由于环境操控模型采取了主动变化状态的方式，是一种主控模型。它的正确答案是C。

一语说明该模型的发挥的主要作用：环境操控模型是一个将所有与一个特定的状态相关的行为封装在一起，并能够被客户端灵活选择，从而使自身的状态转换的一种模型。也可以理解为业务的动态转换，导致自身状态转换的

连续发生的模型。

应用架构图：

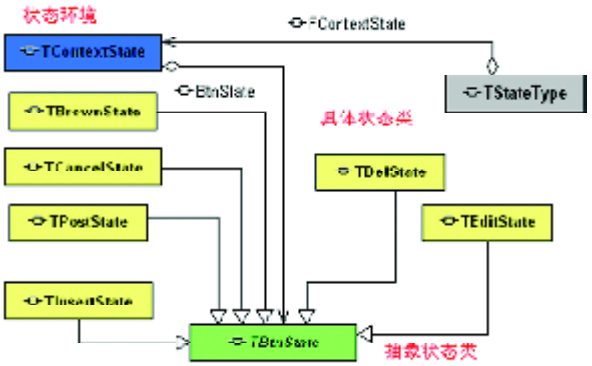


图 2

其中：

抽象状态类（TState）：TBtnState。

具体状态类（ConcreteState）：TInsertState、TDeleteState、TPostState等等。

状态环境（TContext）：TContextState。

实现代码：

TContextState是一个状态环境，是客户端的主要接口，客户端可以为TContextState分配一个具体的状态对象实例，一旦分配成功，客户端就可以通过此管理器和具体的状态对象进行交互。所以TContextState的BtnState属性可以动态改变，并且可以使其引用的具体状态类的实例进行改变。它的定义如下：

```
TContextState = class(TObject)
  private
    FBtnState: TBtnState;
    constructor Create(ABtnState: TBtnState);
    destructor Destroy; override;
    procedure SetBtnState(Value: TBtnState);
  //客户端通过此函数进行状态请求
    procedure StateRequest(ADBBtnMediator:TDBBtnMediator);
public
    property BtnState: TBtnState read FBtnState write SetBtnState;
end;//环境的内部状态
```

BtnState属性代表着TContextState的内部状态，它可以去推动具体的状态类进行实例处理。

StateRequest是进行状态请求的具体函数，它担当了由客户端进行状态实例请求的一个传递者。当BtnState发生变化时，委托StateRequest进行状态传递，但是它本身的代码却不发生变化，真正发生变化的是TBtnState的派生类。它的代码如下：

```
procedure TContextState.StateRequest(ADBBtnMediator:TDBBtnMediator);
  begin
    BtnState.StateHandle(self,ADBBtnMediator);
end;//状态对象处理的句柄
```

抽象状态类TBtnState是一个控制数据操作状态的父类，封装一个TContextState类执行的特定行为，它也担当了请求具体业务的状态实例。声明如下：

```
Procedure StateHandle(Context: TContextState;
                      ADBBtnMediator:TDBBtnMediator);
          virtual; abstract;//状态对象处理的句柄
```

由于此函数是一个虚抽象函数，所有继承于TBtnState的派生类都必须实现此函数，其中TInsertState类便是它的派生类之一，它代表了数据的增加状态，也代表了请求具体业务状态的实现子类。我们以TInsertState为例来说明具体的状态类，它的声明如下：

```
TInsertState = class(TBtnState)//继承
private
  constructor Create; overload;
  constructor Create(AState:Boolean;ADBBtnMediator:TDBBtnMediator);
overload;
  procedure StateHandle(Context: TContextState;
                        ADBBtnMediator:TDBBtnMediator);override;
  //实现状态的处理句柄
end;
其中StateHandle是具体实现状态处理句柄的函数实现如下：
Context.BtnState:=TInsertState.Create(false,ADBBtnMediator);
//状态的创建
```

此代码的实现改变了TContextState的BtnState属性，并在此函数中达到了创建了TInsertState状态的目的，当然也可以在这里进行其它状态的创建，以满足业务中需要达到状态变化的需求。

状态模式能够把特定的状态行为分隔开来，并把一些继承于同一抽象类的派生类进行统一管理。所以通过状态模式可以很容易地使客户端增加或转换新的状态，而不会影响到其它的类，使系统具备了灵活地改变状态而又不需要使用大量开关语句来汇集方法的能力。

导航逻辑模型是被动接受环境操控模型请求的第一个模型，它用一组按钮实现了一个因操作状态改变而数据集的操作状态发生变化的情景，类似于数据导航栏。例如：当前数据操作状态可以从浏览状态转变为增加状态，保存状态，修改状态等等，随即系统将把一些符合当前状态的

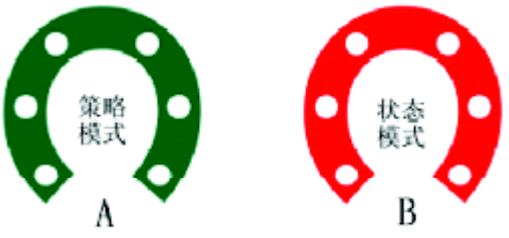


图 3

按钮进行动态 Enabled 切换以满足屏蔽按钮的作用。

图 3 是其备选答案。

通过前文对两种模式的讨论，我想读者很容易地选出正确答案。选题 A 是正确的。它是一种算法，是一种被环境操控模型所选择的被动模型。

应用架构图：

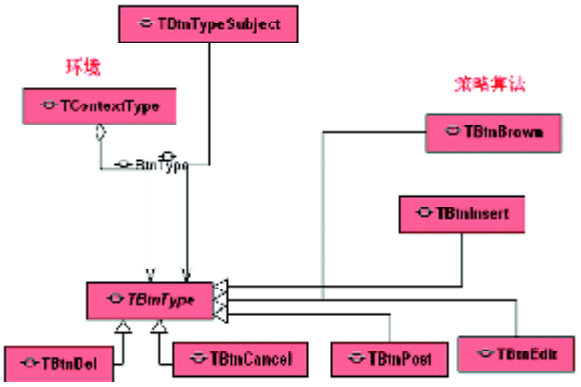


图 4

抽象策略（Strategy）：TBtnType。

具体策略（ConcreteStrategy）：TBtnDel，TBtnCancel，TBtnEdit，TBtnInsert 等等。

环境（Context）：TContextType

实现代码：

模式首先通过 TContextType 类传递客户端所需要的策略类来进行相应的策略计算和策略选择。这样客户端只需和 TContextType 进行交互就能购选择封装在 TBtnType 中的策略类。TContextType 的定义如下：

```
TContextType = class(TObject)
  private
    FBtnType: TBtnType; // 引用数据操作类
    constructor Create(ABtnType:TBtnType);
    destructor Destroy; override;
    function ContextInterface: TAdvAction;
    procedure SetBtnType(Value: TBtnType);
  public
    property BtnType: TBtnType read FBtnType write SetBtnType;
    // 客户端通过此属性传递策略类
  end;
```

TBtnType 是一个抽象策略类类，所有继承于此类的派生类都必须实现其抽象方法。定义如下：

```
TBtnType = class(TObject)
  private
    FOnAdvExecute: TNotifyEvent;
    procedure DoOnAdvExecute(Sender: TObject); virtual;
    procedure OptDataSet(Sender: TObject); virtual; abstract;
    // 策略模式的算法实现
     procedure SetOnAdvExecute(Value: TNotifyEvent); virtual;
abstract; // 自定义事件的实现
  public
    procedure SetDataSet;
  published
    property OnAdvExecute: TNotifyEvent read FOnAdvExecute write
          SetOnAdvExecute; // 自定义事件
  end;
```

其中 TBtnInsert 是 TBtnType 的一个派生类，它是数据操作模型中的 TInsertState 状态类的实例，它的策略算法定义如下：

```
procedure TBtnInsert.OptDataSet(Sender: TObject);
begin
FDBBtnMediator.FBtnEventContainer.OnInsertClick(FEntityName);
// 按钮事件的激活
 FDBBtnMediator.DataEngineMediator.EntityFactory.GetEntityObject
(FEntityName).DataSetOperation.InsertRecord;
// 具体策略的算法，包括增加事件的产生和操作数据集
end;
```

通过以上论述，我们看到客户端只需要通过 TContextType 类传递将要操作的策略类，并与 TBtnType 抽象类进行互动访问就可以很轻松地选择此类下的不同策略算法，如 TBtnInsert、TBtnDelete 等等。这种极易选择产品的方法也提高了导航逻辑模型的灵活性。

我们的两种主要模型都已经介绍完毕，通过对它们的论述，读者已经亲身体验了用 GoF 模式进行架构的思路和方法，我想我的文章若能给读者带来一丝启发，我就为之欣慰了。我还将在下篇文章中对其他的三种模型和接口进行分析，敬请关注。

参考资料

1、关于 UI

UI 框架只是我们整体框架的一小部分，但他却发挥着重要的作用。他的完全实现使整体框架在界面的控制上有一个整体的提升，使由此框架所组成的信息化系统可以很好的被二次维护。该 UI 模型已经在某大型钢厂的信息化和某化工厂的信息化中得到了充分、彻底地运用。

2、读者可以从如下网站找到和设计模式相关的更多内容：

http://hillside.net/patterns/

http://www.csc.calpoly.edu/~dbutler/tutorials/winter96/patterns/

http://www.patterndepot.com/put/8/JavaPatterns.htm

http://www.jdon.com/designpatterns/patternsimportant.htm

作者介绍：卫昆，系统架构师，项目经理。1996 年来着迷于计算机软件开发，从最早的过程开发到现在的 MDA 模型驱动架构，兢兢业业，不敢有一点松懈。几年来，始终恪守“天行健，君子以自强不息。”的信念在软件业里打拼。现致力于管理咨询方面的研究与探索。

■ 责任编辑：崔轶（cuiyi@csdn.net）

基于 Web 方式的 CMS 设计与实现

文 / 赵明超

作者在2004年的年底，受命根据企业、组织内容管理需求的共性，采用面向对象的Java技术，如Servlet、JavaBean、JSP、JDBC、Taglib，开发一种与平台无关、结构良好、拓展性强、可以解决实际问题的CMS系统。现在回过头来总结系统的整个设计和开发过程，作者愿意与各位读者交流一下其中的经验和体会。

内容管理，英文名为：Content Management，是目前业内的一项热门技术和应用。对于这项新兴的技术，业内尚未有一个统一的定义。一般的定义为：内容管理是企业、组织将非结构化信息出版到Intranet，Extranet和ITE(Internet Trading Exchanges)，从而使用户可以方便地检索、使用、分析和共享的处理过程。

内容管理由两部分组成，“内容”强调对象，而“管理”则强调方法。内容实质上就是任何类型的数字信息的结合体，可以是文本、图形图像、Web页面、业务文档、数据库表单、视频、声音文件等。管理就是施加在“内容”对象上的一系列处理过程，包括收集、确认、批准、整理、定位、转换、分发、更新、存档等，目的是为了使“内容”能够在正确的时间、以正确的形式传递到正确的地点和人。内容管理系统就是能够支撑内容管理的一种工具或一套工具的组合。

Gartner Group在2000年9月份的一个调查显示：平均每个知识员工每个星期需要花费8个小时在不增值的相关文档处理任务上，包括文档的创建、寻找、整理等。而Forrester Research也给出了一个统计数据，其调查显示内容量在以每年200%的速度增长。一方面是低下的内容处理效率，另一方面是不断膨胀的内容量。这简单的事实对比已经隐含了企业的一个迫切需求：我们需要有效地管理我们的内容。

系统结构

本CMS系统包含系统管理模块、内容管理模块和客户访问模块。其中内容管理模块是整个系统的核心，它负责信息的加工、处理、发布过程；系统管理模块则主要是体现了权限的设定；客户端访问模块是整个系统运行的主要目的，即提供加工好了信息给客户。三个模块良好的互动，有机的集成为一个应用。

系统功能结构

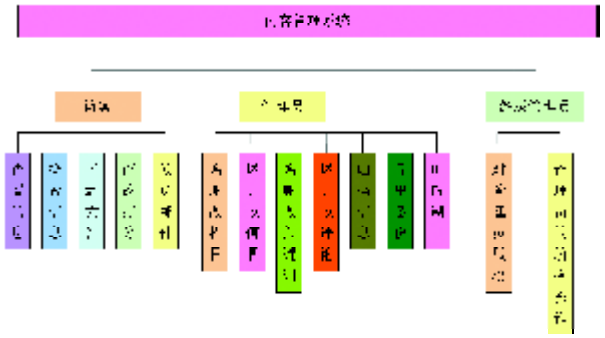
系统根据使用对象的不同将用户角色主要分成三种：访客、管理员和超级管理员。

管理员负责整个系统的数据来源，他们担负着信息收集、整理、加工和分发的重任，是信息的生产或加工者。其具体职责包括添加、删除、修改栏目、信息、关键词和评论，将信息归档，导出信息备份，应用IP限定。管理员可能只能管理某一个或几个栏目，这取决于超级管理员赋予的权限。

访客则可以访问网站，浏览、查看、检索信息，下载文件，发表评论、打分，将信息Email给他人。CMS系统的最终目的就是将经过加工的信息以某种方式提供给用户，所以访客就是最终的信息消费者，也是系统服务的主要对象。

超级管理员除了具备访客和管理员的一切权利之外，还可以赋予或剥夺管理员的权限制定需要限制的IP段。

功能结构如下图所示：



系统逻辑结构

从逻辑上讲，本系统分为四个层次，分别为表示层、表示逻辑层、业务逻辑层、持久层。如下图所示：

表示层（Presentation）即客户端的浏览器通过HTML向最终用户展示被请求的信息。

表示逻辑层(presentation Logic）在一个或多个Web服务器构建的空间里运行，使用JSP，Taglib，Servlet完成动态内容显示和流程逻辑控制，使用xml+xslt 生成HTML文件。

业务逻辑层(Business Logic）由若干运行在Web服务器下的JavaBean组成，这些组件是可以重复利用的，可以不做任何修改就移植到其他的服务器上。而JavaBean对数据库的访问是通过JDBC来实现。

持久层(Persistence）由关系型数据库系统和xml组成。

本系统在实现过程中力图遵循J2EE架构标准，把表现逻辑、业务逻辑和后端的数据服务分开，保持每层的相对独立性，松耦合度。每层只提供特定的服务，任何一层只要其提供的接口不变，在实现上发生什么变化对其他层没有影响或者影响很小。使各层功能变得简单、专一、容易实现，而且具有更好的可维护性和可拓展性。

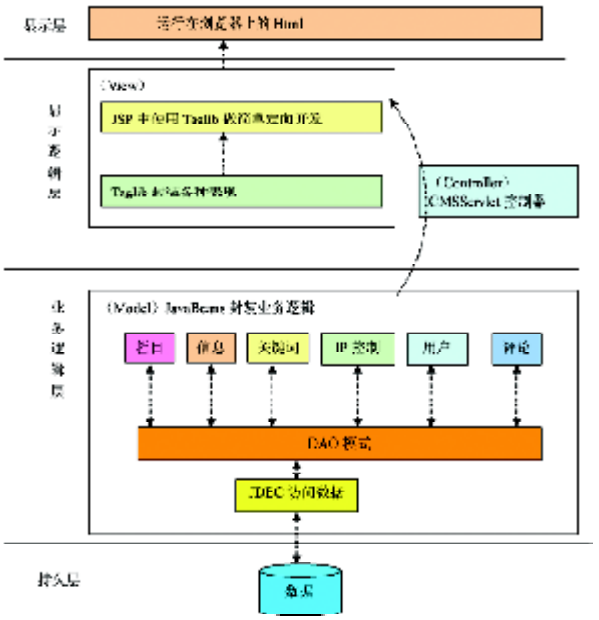
针对不同的需求和不同的表现形式，只需在既有系统的基础上通过设定参数决定需要使用的模块，再加上Taglib组合开发出多样的表现形式，尽量少写代码(见右栏上图)。

设计与实现

CMS 系统的功能模块设计

CMS系统模型是由一系列接口定义清晰的独立模块组成。

- 控制模块：负责接受客户端的请求、调用业务逻辑操作、将相应的视图返回给客户。
- 信息管理模块：负责信息添加、修改、删除，和栏目的添加、修改、删除；信息的导出备份。
- 关键词管理模块：负责对关键词的添加删除。



- 评论管理模块：负责对评论打分的管理;为可选模块。
- IP 控制管理模块：负责建立、删除、修改 IP 范围，以便在管理信息的时候能够选用；为可选模块。
- 用户管理模块：添加、删除、修改用户信息，赋予权限。

这些模块被封装成为独立性和内聚性较强的模块。可选模块通过存储在 xml 文件中的系统设置来决定在系统启动的时候是否加载。

CMS 系统的数据库设计

CMS 系统主要设计了信息表、栏目表、关键词表、评价表、IP 块表和IP规则表。这是应用中很重要的一部分，它将直接影响到系统的长期有效性和产品化程序。

信息表中存储了信息的数据。在信息表中设计了名称和标题两个字段，前者主要用在标签上。对使用标签库（Taglib）的开发者来说，名称比一个单纯的ID数字要有好一些；后者则表示信息的标题（Subject）。

对于可能出现的各种属性和状态（譬如发布、审核、失效），我们使用了一个长整型的数字来代表。这个数字的每个“位”代表一个意思。然后通过“与”和“或”的运算获取相应的设置情况。譬如我们做以下规定：

1 表示该信息已经审核通过

2 表示该信息已经发布

4 表示该信息被放置到了回收站

在默认的情况下，每条新加入的信息的状态属性为0。管理员审核通过信息之后，状态“或”上1，点击发布之后状态“或”上2。可以有这样的代码：

```
private static final long PASS = 1;
private long state = 0;
public void pass(){
  state |= PASS;
}
```

在检查是否为已经发布的时候，则采用“ 与 ”操作，可以有这样的代码：

```
public boolean isPassed(){
    return (state&PASS)>0;
  }
```

栏目表则存储了栏目相关的数据，譬如ID、标题、属性设置，以及栏目在二叉树中的左节点和右节点。在建立树的时候首先通过左右节点将线型数据转化为二叉树，然后再将二叉树转换为一棵树。

关键词表存储了自定义的所有关键词；评价表存储了对信息的评价和打分情况；IP块表存储自定义的IP段；IP规则表则是记录信息应用IP情况，是一个多对多的关系表。

基于MVC模式的Web层实现

MVC（模型 - 视图 - 控制）设计模式将Web层组件分为三类对象：分别代表数据的模型（Model）对象，显示模型的视图（View）对象，以及响应用户输入、处理业务流程的控制器（Controller）对象。每类对象都有明确的责任：模型（M）不能包含特定于视图的代码；视图（V）不能包含控制代码或者业务逻辑代码，只能用来显示数据；控制器（C）负责创建和更新模型，而不依赖于任何特定的视图实现。MVC模式将表现层逻辑与业务逻辑分离，提供了一个Web界面和业务对象清晰交互的理想模式。（《程序员》2004合订本（下）.P220 - -《J2EE企业架构的参考模型和实现策略》）

本系统设计了一个Servlet作为控制器（Controller）作为整个应用的入口点，并在标准的Web部署描述符文件“ web.xml ”中将请求URL全都映射到该控制器上；使用Taglib来分离表现逻辑和业务逻辑；采用数据绑定功能，将HTTP请求参数透明地更新到JavaBean（命令对象）的属性中去，如信息、栏目、关键词。系统运行中客户端的请求首先被控制器截取，这个控制器初始化请求，执行前端到后端的处理（front - end processing），包括身份审核、授权、登录，业务逻辑处理。一旦请求完成，控制器直接将请求指向到合适的视图（View）。

经过这样的处理，JSP页面只是起到表现作用，与业务逻辑通过控制器分开，便于系统的复用和更新拓展。如图2所示。



图3：MVC模式示意图（原图引自○ 1Chapter3.3）

采用基于JDBC的DAO模式隔离数据层

数据访问对象（Data Access Object）模式，简称DAO模式。DAO模式的目的是降低一个应用中业务逻辑和数据访问逻辑的耦合度。当使用永久层框架的时候，这个模式可以帮助降低逻辑业务和框架之间的耦合度。第二个目标是允许永久层框架可以很方便的被另一个替换而无须修改业务逻辑。

以信息类为例，设计一个信息DAO工厂类、一个信息DTO类，一个DAO数据访问接口，和若干个对数据访问接口的实现类。其中信息DTO类是对信息数据表的映射，用来是储存数据，只存在getter和setter方法。在DAO数据访问接口中定义访问数据库方法，通过DTO传递数据。针对不同的DBMS有对DAO接口不同的实现，通过工厂类可以生产特定的数据库访问实现。如图4所示。

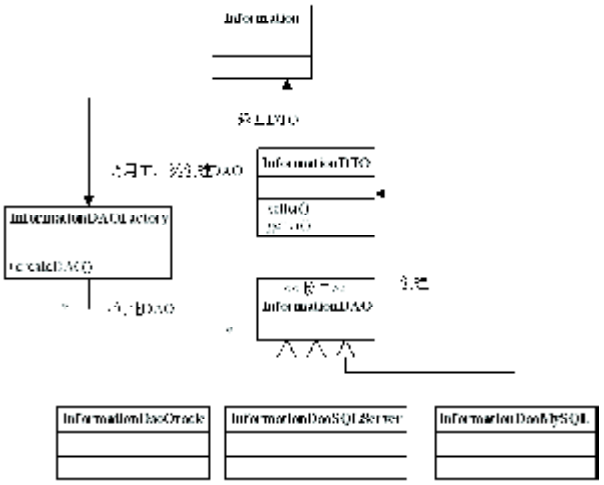


图4：DAO模式示意图

通过这样一种操作模式我们就实现了对数据源的松耦合，可以方便的替换为不同的数据源或永久层框架。

信息管理模块设计与实现

CMS系统的中心是信息，这里重点介绍信息处理模块。

信息处理

我们分别设计了信息类（Information）来代表各种信

息，信息内容类（InformationBody）代表要处理的信息的内容，信息管理类（InformationManager）来负责同信息相关的操作。

信息类（Information）主要是存储信息数据（譬如ID、标题、内容、设置等），和一些同信息对象自身的一些操作，譬如获取内容、加载数据。而信息管理类（InformationManager）负责管理信息，包括添加、删除、修改、查询以及获取一个信息实例等。它被设计成为单例模式类，也就是说在系统运行期间整个JVM上只有一个信息管理类（InformationManager）存在，任何对信息管理类的请求得到的都是同一个对象。

之所以将信息内容单独设计成一个类是因为，我们这里将各种信息处理后存储为大字符串，如果在信息加载时也加载则会占用大量内存空间。设计成为单独的对象则可以仅仅在需要的时候加载这个大对象，对系统运行性能提高有帮助。而且这种设计方式在开发上要更加简单一些，不会让一个对象承担太多的责任。

栏目处理

类似的，在处理栏目的时候我们也设计了栏目类（Column）用来存储栏目数据（如ID、栏目标题、栏目设置）、栏目树类（ColumnTree）（用来存储栏目树状的层级结构，获取节点信息）和栏目管理类（ColumManager）（负责同栏目相关的操作，譬如添加、删除、修改和获取一个栏目实例）。同样栏目管理类也采用了单例模式。

系统在运行期间操作最频繁的其实就是信息和栏目，对二者我们设计了两个管理类，一方面对于系统的性能帮助很大，另一方面也减轻了信息和栏目的负担，整个设计清晰明了。

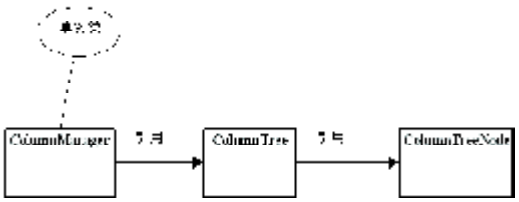


图 5

在这里，我们令每条信息都属于某个栏目，不存在没有栏目的信息。栏目对象设计成为树状结构，单例的栏目管理类中维持一个对栏目数的引用，这样栏目树也无需每次去加载，避免生成树的时候对资源的占用。只有在这棵树的节点发生变化的时候才重新加载，比如改变栏目的标题或者将一个栏目转移到其他的栏目下。

信息和栏目的表现形式处理

对于信息和栏目不同的标题表现形式，比如一个图片新闻在可能以图片的形式展示其标题，我们设计一个ViewType接口。这个接口只提供三个方法：获取类型（如image、rtf）、获取视图（如图片显示方式）、获取类型的中文名称。所有系统支持的媒体类型都实现这个接口。

在JSP调用标签库显示信息的时候，首先根据信息的表现形式的不同生成ViewType实现类，然后调用其获取视图方法。

我们通过：表现接口（ViewType）+表现实现类（ViewTypeImpl）+标签库（Taglib）的方式，即可解决不同信息类型显示的问题。

对于导出信息部分的处理

在这里，考虑到用户友好性和客户端应用程序的兼容性，我们采用输出为Html文件的方式。为此我们使用xml + xslt的方式，将信息在服务器端生成html。如果是某个栏目的输出就组合成恰当的层次结构，然后压缩输出。

为此设计一个信息流类（InformationStream）、一个栏目流类（ColumnStream）、一个导出Servlet（OutputServlet）。信息流类负责信息内容的检索，生成xml文档，再根据xslt样式表生成表现丰富的Html文档。栏目流类主要处理生成栏目结构，引用信息流类处理其下的信息。在做导出操作的时候，导出Servlet将导出的Html格式信息压缩到输出流（OutStream）中，输出给最终用户。

在信息流类中使用了Jdom处理xml的生成，以及结合xslt生出Html。

```
// 将数据库中检索到的信息转化为 xml 文档
// 初始化根节点
Element root = new Element("information");
Document doc = new Document();
// 添加根节点
doc.setRootElement(root);
// 添加内容
root.addContent(new Element("subject").setText(info.getSubject()));
root.addContent(new Element("keywords").setText(
                                        info.getKeywords()));
root.addContent(new Element("body").addContent(
                                new CDATA(getBody(info))));
root.addContent(new Element("author").addContent(
                                          info.getAuthor()));
// root.addContent(……);
// 获得一个 xml输出处理对象
XMLOutputter outputter = new XMLOutputter();

Format format = outputter.getFormat();
format.setEncoding("GBK");
format.setIndent("  ");
```

```
format.setLineSeparator("\n");
outputter.setFormat(format);

ByteArrayOutputStream out = new ByteArrayOutputStream();
// 将 xml转化为 Html
try {
    Document htmlDoc = transform(doc, xslPath);
    outputter.output(htmlDoc, out);
} catch (JDOMException e) {
    e.printStackTrace();
}
// 写入压缩流
writeStream(zos, zipPath, new String(out.toByteArray()));

// 将 xml结合 xslt 生成 Html的转换方法
public Document transform(Document doc, String xslPath)
            throws IOException, JDOMException
{
    try {
        // 获得一个转化实例
        Transformer transformer = TransformerFactory.newInstan()
                                  .newTransformer(
                                  new StreamSource(xslPath));
        // 获得输入和输出
        JDOMSource in = new JDOMSource(doc);
        JDOMResult out = new JDOMResult();
        transformer.transform(in, out);

        return out.getDocument();
    }
    catch (TransformerException e) {
        throw new
            JDOMException("XSLT Transformation failed", e);
    }
}
```

IP 控制模块

所谓IP控制主要是指信息在客户端被浏览时，可以限定在指定的IP范围之内。系统管理员首先需要制定一个IP块（IPBlock）列表。对于每个IP块都需要指定子网ID和子网掩码，也就是指定IP范围。

建立了列表之后，管理员就可以根据实际情况将IP限定应用到特定栏目或者特定信息上。

在jsp中调用Taglib显示信息或者栏目的时候，即可以通过计算用户的IP地址与管理员设定的IP范围是否吻合来确定是否允许浏览。

显示层的标签库

位于表现层的标签库（Taglib）包括以下标签：

- 信息列表标签：负责信息列表显示，可嵌套使用。
- 信息详情标签：负责信息详情展示，处理不同的标题和内容的显示方式，譬如文字、图形图像、Flash动画。
- 栏目列表标签：负责栏目列表现是，可嵌套使用。
- 栏目详情标签：负责栏目详情展示，处理不同标题的显示方式，譬如文字、图形图像、Flash动画。
- 关键词连接标签：当根据关键词找到相关信息。
- 信息检索标签：根据检索词查找相关信息，并根据相关度排列。
- 评论标签：负责打分评价功能。
- 发送Email标签：负责发送Email。

在标签中，我们封装了表现逻辑，进一步减轻了一般开发人员的负担。基本上针对每个特定的对象和特定的功能我们都设计了标签，通过对这些标签或简单或复杂的嵌套、组合、排列，再加上自定义的样式表单，就提供给用户足够丰富的表现方式。

结论

本CMS系统采用MVC模式分开了表现和逻辑，以DAO模式脱离对特定数据源的依赖，采用标签库令表现层开发人员无需掌握复杂的逻辑就可以开发出具备足够复杂表现样式的界面，以xml + xslt方式解决了导出数据的友好性和客户端的兼容性，并提供了一些基本的、可拓展的功能模块。目前该系统在近三十家高校使用中。

参考资料

1、Chuck Cavaness, Programming Jakarta Struts, O'Reilly, November 2002

2、Jason Hunter with William Crawford, Java Servlet Programming, 2000 O'Reilly

3、天道酬勤, 门户中的内容管理系统, http://blog.csdn.net/andiyang/archive/2005/01/28/272456.aspx

4、 Bruce Eckel, Java编程思想（英文版，第三版），机械工业出版社，2004年1月第一版

5、读者可以从下面链接找到和CMS相关的更多内容

http://www.cmswatch.com/CMS/

http://www.zope.org/Products/CMF/

http://www.steptwo.com.au/papers/kmc_evaluate/



关于CMS最完整和权威的参考书籍

■ 责任编辑：崔轶（cuiyi@csdn.net）

趣谈 Java 语言 10 年

文 / ajoo

十年，十年了。少年子弟江湖老，花开花落又一春。一眨眼，当年不起眼儿的小屁孩儿都已变成笑颜如花的美少女；而当年玉树临风，指点江山，潇洒倜傥者如鄙人者，却已经精神矍铄，鹤发童颜了。不小心瞟一眼编程语言，咦？这不是Java吗？几年下来，可真是出息了！

坦白说，Java这十年取得了惊人的成功。其实，这世界上的编程语言至少有上百种，运行效率高过Java的，用起来比Java简单的，类型系统比Java严谨安全的，语言设计更典雅优美的，表达能力更强的，语法设计更性感摇滚的，并不是没有。但你气愤也好，嫉妒也罢，Java这个幸运儿就在这么短短的十年间扶摇直上，小人得志。不说一支独秀，腕儿也是大了去了。

Java发达，到底凭什么？是撞了大运，让天上落下馅饼直接砸脑门子上？

要说运气，这一定是有的。但是，根据我总感觉，Java的成功相当一部分的可以归功于一个字：平衡（balance）。

评价标准

评价一个编程语言，一般有这么几个标准：

1．稳定性。用这个语言写出的程序是否容易出错？是否容易崩溃？计算机是严谨的，而人是容易晕菜的；计算机算1+1多少次也都是2，而人就没准儿算出个1来。所以一个好的编程语言应该能够未雨绸缪，把因为人的因素犯错误的机会降低到最小。而一旦真正出了错误，也要尽量减小错误的损害程度。

2．简单性。爱因斯坦说过，上帝是喜欢简单的。这个简单还分几个内涵：学习起来简单，语言设计实现简单，使用简单。

3．效率。虽然说硬件越来越便宜，CPU越来越快，但是程序的效率问题仍然远远没有退出历史舞台。如果语言面向的是实际工作中的应用，那么效率的问题至少到现在还不容忽视。

4．平台无关性。能够把自己写的程序在任意硬件和操作系统上运行自然是美事。为啥英语牛？不是因为英语多么美，多么科学，而是因为它是国际通用语言。

5．使用方便，功能强大。甭管是用库，还是用语言本身的设施，一些常用的功能，如persistence，concurrency，常用数据结构和算法，网络等的支持如何，是几乎关系一个语言成功与否的关键。库者，死生之地，存亡之道，不可不查也。

6．语言表达能力。也就是灵活性。这里提的表达能力，是指一般意义上的相当的程序复杂度的情况下，谁能表达更灵活的需求。

对号入座，我们来看看Java是怎么抖起来的。

跨平台

Java开始设计的时候，第一个卖点就是平台无关性。所谓“write-once-run-everywhere”。今天看，虽然因为多线程和一些库的原因（比如GUI）让这个承诺打了折扣，但是Java语言的核心的设计还是真正体现了平台无关的思想。这个跨平台的能力，从Java诞生到现在，几乎没有什么变化，但是它毫无疑问是Java成功的最重的一个砝码。多少企业选择Java的最主要的原因就是它的跨平台能力。因此而付出的一些代价：比如，不能利用操作系统的线程调度，比如不能直接操作物理内存等，都是在这个伟大目标面前必须牺牲的马前卒。凡事不能碗里的锅里的都占着。当鱼和熊掌不可兼得的时候，Java就必须用平衡裁减掉那些相对次要的目标。

稳定性

Java在设计的时候，鉴于很多用其它语言设计出来的程序的稳定性不高，其设计思想就是稳定压倒一切。

我们来看看它为了稳定都做了些什么：

1．更严格的类型系统。相比于当时的C++：

a 字符，字符串都作为单独的类型，避免因为混淆出错。

b 单独的布尔类型。禁止boolean而整数之间的隐式转换。这让if(a=b)这种在C/C++中常犯的错误遁迹无踪。

c 取消了无符号数类型，也就取消了有符号数和无符号数之间因转换而出现的错误。作为更安全的替代品，引入

了>>>, <<<这两个无符号移位操作符。

d 检查型异常。所有抛出的非RuntimeException的异常都要用throws明确声明。这些异常,要不被调用者继续声明抛出,要不被catch住。虽然经过近十年的实践,一些人发现这东西似乎有点看上去很美的意思。Java在这个问题上的权衡,也许错了,也许对了,目前下结论,也许还为时过早。不过它在保证大规模的程序开发中的程序稳定性上还是有功劳的。

e 在Java 1.5中增加了泛型。本来的Collection Framework只处理Object,需要用cast才能把从一个Collection中拿到的对象转换位期待的类型。泛型的引入很大程度上解决了cast造成的潜在的类型匹配的问题。当然,Generics大大增加了语言的复杂度,背离了Java一贯坚持的简单路线。在这个问题上,Java在稳定性和简单性之间再一次作出了权衡,是非功过,也许还要等几年才能见分晓。

f Java 1.5同时也加入了类型安全的枚举。

2.动态检查。虽然Java的类型系统比较严格,但是并不是100%强类型的系统。通过静态类型检查的程序仍然可能在动态的时候发生类型匹配错误和其它错误。通过强制性的动态检查,Java再一次把稳定性放在了权衡的首位,运行效率被一定程度地牺牲了。

a Java的数组支持协变,也就是说,String[]也同时可以用作Object[]。你可能觉得这看上去很好啊。是啊,可惜有时候直觉是不可靠的。我们日常生活中对很多概念的直觉来自于它们不会变化的假设。如果Object[]不改变,那么把String[]当作Object[]并没有问题。但是,看下面的代码:

```
String[] names = new String[]{"a"};
Object[] aliases = names;
aliases[0] = new Integer(1);
String name = names[0];//通!爆炸了。
```

这种类型系统的漏洞就只能靠运行时的类型检查来补救了

b 强制类型转换可能造成错误。

c 对数组元素的存取可能造成下标越界。

d 对象引用可能为null。而对null调用任何method或者field都是非法的。

3.禁止直接操作内存和指针。用对象引用来代替指针。没有了指针算术,access violation错误被彻底杜绝了。

4.垃圾收集。

a GC的出现让“释放内存”永远成为了回忆。没有了释放内存的需要,也就自然不会有错误释放内存,一块内存释放两次这些错误了。

5.强制初始化。

a 所有的变量,即使没有被显式初始化,也会被缺省地初始化。引用会被初始化成null,整数会被初始化成0等。这保证了不会有非法引用的隐患。

6.异常时候的Stack trace。

a 一旦真正出现了异常,造成这个异常的整个调用栈信息都可以被打印出来。这大大方便了错误定位。

b 在Java 1.4中增加了exception chaining的内建支持。当一个异常是其它异常引起的时候,所有直接或者间接引起错误的异常的stack trace都会被打印出来,使得Debug更加容易

7.try-finally机制保证关键的资源能够被即时清理。

对一些关键资源,比如数据库连接,网络连接等,即时地关闭这些资源可以保证在应用服务器中运行的程序不会因为异常的发生而影响到其它正常运行的程序。Java提供的try-finally机制保证了对这些资源的清理不会因为异常的出现而受到干扰。

8.Java 1.4加入了assert的功能。这个assert用来表示在程序的某些地方一些条件必然成立。比如著名的契约式设计里面的前置条件、后置条件以及Invariant。

简单性

Java的设计充分考虑到语言简单的重要性,尤其注重让学习这个语言的人能够一目了然。

1.单根结构。所有对象都从Object继承下来。

2.对象都是引用语义。也就是说,在Object ref = …;中,ref就是一个指向一个储存在堆中的引用。这个引用被存在栈中,并且当返回或者传入某个函数的时候被复制。

3.所有函数调用都是call-by-value。不支持call-by-ref,call-by-need等各种函数调用约定。当然,因为对象变量的引用语义,对Java是否是call-by-value仍然在很多初学者中造成了疑惑。其实,一言以蔽之,函数调用都是call-by-value,只不过对对象来说,变量本身的value就是引用。

4.所有非私有函数都是可以被子类override的,也就是都是virtual的。

5.wait、notify被直接放入Object中。

a 其实,如果选择把wait()、notify()单独放在一个Lock类中也不是不可以。而且,因为可以把处理锁需要的数据结构从Object中剥离出来,反而会得到更好的效率。

b 但是,现在的这种设计,让任何一个Object都可以被同步,都可以wait()、notify()。于是学习更加容易了。语法上,那个声明在method前面的“synchronized”的语法糖也就成为可能。

类似还有很多,这里就不赘述了。

效率

可以说Java并不是一个效率至上的语言,在前面的分析中我们已经看到效率在和其它Java认为更重要的目标的权衡下被牺牲了。

但是,Java的设计也并不是无视效率。实际上,Java为了对效率的妥协,就放弃了“万物皆对象”的美丽光环。

1．Java的原始类型——int、short、char都不是对象，它们不象对象一样遵循引用语义，而是遵循值语义。也就是说，对一个int类型变量的赋指，是真正改变了这个整数，而不是仅仅改变指针的指向。这样做的好处是明显的，这些原始类型不占用堆空间，而是直接分配在开销非常低的栈中（甚至可能是寄存器里面）。垃圾收集器的负担也得以减轻。但是，坏处也同样是明显的，语言内部出现了原始类型和对象的不统一。一定程度上牺牲了简单性和一致性。有int、char还要有Integer、Character，有时候还真是不爽得很啊！

2．Java 1.4中加入的assert也是基于对效率的考虑。本来，如果不关注效率，一个小小的函数就可以解决问题，比如：

```
void assert(boolean cond, Object msg){
    if(isAssertEnabled &&!cond) throw new
AssertionError(" " +msg);
}
assert(list.contains(x), "x should be in the
list");
```

这个assert在isAssertEnabled为false的时候就不会抛出任何异常。

但问题在于，Java的函数都是call-by-value的，也就是说，无论如何，这个list.contains(x)都会运行，而这个动作也许是比较昂贵的。

要达到让assert函数在isAssertEnabled为false的时候不对list.constains(x)这个表达式求值，就必须要用call-by-need或者call-by-name或者宏。而这些对语言的复杂性的影响就大了去了。

权衡轻重，Java 1.4里直接把assert加进来作为一个语言提供的特殊设施，保证了在assert被disable时，被assert的表达式本身就不会被处理。

从语言的美感来说，这么做实在不漂亮。这个assert就是一个硬邦邦塞进Java里的特殊规则。什么道理也没有。

更何况，assert的表达式可能是带有副作用的。这就造成了潜在的debug和release版本的程序行为不一致的问题。

不过呢，反正语言这么定义了，这就是天大的道理，上天注定的最大嘛。

使用的方便程度

Java拥有强大的类库，而且这些年来无论库还是语言方面的改善都是很大的。

1．字符串类型对加号的重载。让习惯了运算符重载的我们多少找到点安慰。虽然不是普遍意义上的运算符重载，而且还是一个特殊规则，但是，毕竟用起来方便了。

2．内建的的对象克隆机制简化了深度复制对象的工作。

3．垃圾收集。管杀不管埋，吃饭不给钱。

4．丰富的基础类库，比如网络，I/O等。工欲善其事，必先利其器！

5．Java 1.1增加了对串行化的支持。任意一个Java对象都可以在不用增加任何代码的情况下被写到文件等I/O设备中，或者从中读取出来动态重新生成对象。持久化向来是程序设计中的一大难题，Java提供的这样一个简单的方案真是让人叹为观止。

6．Java 1.1重新设计的Collection Framework几乎涵盖了程序中大多数的数据结构和常见算法。使用也非常简单。

7．Java 1.5中新增加的增强for loop，可变参数，枚举类型等，都提高了Java代码书写的方便性。

8．1.5引入的annotation，或者说元数据，也是一个非常非常有用的特性。应该会成为框架作者们手中的利器。

语言的表达能力

Java的语言的整体框架思路非常清楚：用类来封装数据和行为，用接口来提供模块之间的契约和松耦合，具体的一些模块的实现上还可以用反射的动态能力来弥补多态的不足，比如用Java 1.3开始支持的动态代理来实现一些AOP的思想等。

Java的强类型对灵活性有一定的限制，1.5之前，我们用反射和强制类型转换来绕过这些限制。Java 1.5则通过加入泛型来改善静态类型系统。双管齐下，Java现在还没有Ruby这样的动态语言的灵活性，也没有Haskell这样的静态语言的类型系统的优雅安全，可以说它是一个不完美但是实用的折衷产物。它的广泛流行也证明，Java走的这条中间道路是成功的。

展望

Java发展了十年，到今天，语言上的变化不太大，除了近期的1.5的泛型算是一个大动作以外，多数的改进都是在类库上。当然，这也是自然的。对语言的改动一定要非常慎重。如果让我说说我期望的语言上的改进，我最盼望的会是类似C#那样的委托，或者用C++的黑话讲：函数指针。

现在的Java是通过匿名类来实现类似的目的。但是匿名类语法相当繁琐，代码读起来比较臃肿。还有一些程序员通过反射来自制委托，比如new Delegate(obj, "mtd_name");。这种方法避免了匿名类的缺点，但是失去了静态类型安全性，而且反射和动态代理的效率也比较低。

另外一个非常有趣的动向就是动态语言。近几年随着Python，Ruby的出现，掀起了一股动态语言的热潮。静态类型系统重要性进一步下降，而语言的简洁直接灵活上升到首要的位置。Java虽然是一个传统的静态强类型的语言，但是仍然有Jython、Groovy等动态类型的脚本语言加入到Java平台的阵营中来。

■ 责任编辑：崔轶（cuiyi@csdn.net）

Eclipse RCP 入门

文 / 陈刚

Eclipse的插件机制更是提供了一种令人耳目一新的开发方式。现在国内做Eclipse插件的渐渐多了，并且很多用SWT/JFace和插件方式来开发应用项目中的界面。这时将插件用于应用类项目的就有些不爽：插件要依赖Eclipse运行，而这些项目大部需要程序更象一个传统的应用程序，如果表面上看起和Eclipse一点关系也没有那就最好了。而Eclipse3.0之后便解决了RCP的这个问题。本文作者通过一个开发实例，介绍了一些RCP所特有的东西。

我们这一步将利用Eclipse提供的一些项目模板，来快速创建一个初始的RCP项目结构，后面再在这个基础上不断的修改和补充，最后形成一个标准的RCP程序。当然，如果你不想用模板，而是自己从无到有的创建项目，也是可以的。

具体操作步骤如下：

(1) 选择Eclipse主菜单“File ->New ->Project…”

(2) 在弹出窗口中选择“Plug-in Project”项，单击“Next>”

(3) 填写项目名(Project name)为“myrcp”，单击“Next>”

(4) 在Class Name项，将其修改为“cn.com.chengang.myrcp.MyrcpPlugin”，单击“Next>”

(5) 选择“Hello,World”模板，单击“Finish”完成项目创建。

(6) 这时可能会弹出“Open Associated Perspective？”的提示框，问你是否转到“Plug-in Development”透视图，建议选择“是”。

经过以上操作之后，Eclipse将根据内置的“Hello,World”模板来创建一个插件项目，myrcp在“Package Explorer视图”中的项目结构如图1所示。

在项目中有两个非常重要的配置文件：MANIFEST.MF、plugin.xml。MANIFEST.MF包含一些版本号、开发者、classpath等信息。plugin.xml是插件的入口，Eclipse根据它里面的配置信息来加载插件，在开发初期会频繁编辑此文件，术语叫“设置扩展点”。比如，在Eclipse的增加透视图、视图、主菜单等，都需要在plugin.xml里设置相应扩展点。

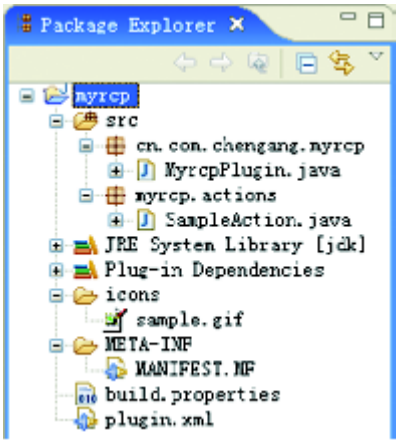


图1：myrcp项目结构图

> 注：过去，插件的主要配置文件是plugin.xml，在eclipse3.0之后才又增加了MANIFEST.MF，并将一些在plugin.xml设置的项，转移到了在MANIFEST.MF里来配置。

修改成为RCP项目

上面创建的项目还仅是一个插件项目，现在我们来将它改造成一个RCP项目，具体操作步骤如下：

(1) 先将我们不会用到的Sample-Action.java和myrcp.actions包删除。MyrcpPlugin.java文件虽然本文实例中不会用到，但它本身比较重要，在实际开发中是经常要用的，所以保留它。

(2) 将plugin.xml的内容修改如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<?eclipse version="3.0"?>
```

```
<plugin>
   <extension
         id="MyApplication"
         point="org.eclipse.core.runtime.applications">
      <application>
         <run class="cn.com.chengang.myrcp.MyApplication"/>
      </application>
   </extension>
</plugin>
```

说明：

- <extension>是<plugin>的一个子项，它表示一个扩展点的设置。
- id —扩展点的标识号
- point —指设置的是什么类型的扩展点。
- run class —指向 RCP 的入口程序 MyApplication（后面将创建它）

（3）创建 MyApplication.java 文件

```
package cn.com.chengang.myrcp;

import org.eclipse.core.runtime.IPlatformRunnable;
import org.eclipse.swt.widgets.Display;
import org.eclipse.ui.PlatformUI;

public class MyApplication implements IPlatformRunnable {

public Object run(Object args) throws Exception {
        Display display = PlatformUI.createDisplay();
        try {
                int returnCode = PlatformUI.
                        createAndRunWorkbench(display,
                        new MyWorkbenchAdvisor());
                if (returnCode ==
                   PlatformUI.RETURN_RESTART) {
                      return
                       IPlatformRunnable.EXIT_RESTART;
                } else {
                      return IPlatformRunnable.EXIT_OK;
                }
        } finally {
                if (display != null)
                       display.dispose();
        }
  } // End of run
}
```

说明：

MyApplication 必须实现 IPlatformRunnable 接口，它在项目中的位置和 Java 类中 main 方法类似，MyApplication 是整个项目的入口程序。程序中用到了一个 MyWorkbenchAdvisor 类，接下来将创建它。

（4）创建 MyWorkbenchAdvisor.java 文件

```
package cn.com.chengang.myrcp;

import org.eclipse.ui.application.WorkbenchAdvisor;

public class MyWorkbenchAdvisor extends WorkbenchAdvisor {

 public String getInitialWindowPerspectiveId() {
        return null;
 }
}
```

说明：

MyWorkbenchAdvisor 必须继承于抽象类 WorkbenchAdvisor，此类主要负责配置整个工作台。其中 getInitialWindowPerspectiveId 方法是必须实现的，工作台会打开一个透视图，就是通过此方法的返回值来指定。现在返回空值，表示工作台不打开任何透视图。

运行 RCP 程序

选择 Eclipse 主菜单“Run—>Run…”，弹出 Run 设置窗口。在窗口中进行如图 2 所示的设置。

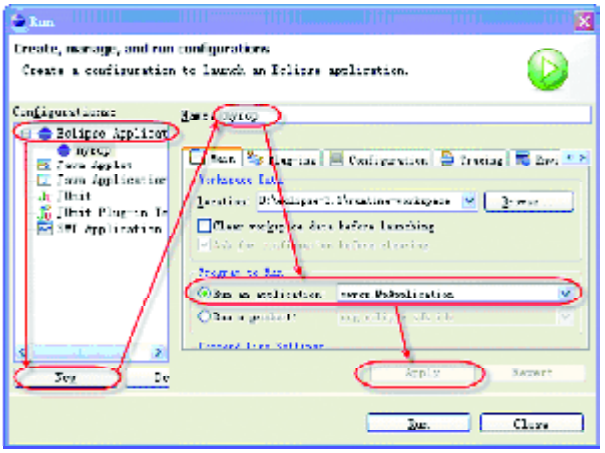


图 2：Run 设置窗

转到“Plug-ins”选项卡，先将“Extenal Plug-ins”下所有的勾选取消，然后单击“Add Required Plug-ing”按钮，Eclipse 会自动将 myrcp 项目依赖的插件都勾选上（如图 3 所示）。

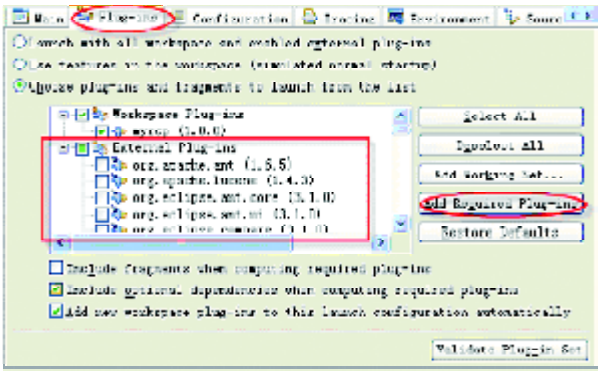


图 3：设置项目运行的依赖项

最后单击“Run”按钮，项目的运行结果是一个空白

的窗体。

加入一个透视图

以上已经完成了一个最简单的RCP程序，它还只是一个空白的窗体。在Eclipse中UI元素都是集中在一个透视图中，因此在一个RCP项目中，通用创建的第一个UI元素就是透视图。给此RCP程序添加一个透视图的方法如下：

（1）创建一个透视图类MyPerspective.java。

```
package cn.com.chengang.myrcp;

import org.eclipse.ui.IPageLayout;
import org.eclipse.ui.IPerspectiveFactory;

public class MyPerspective implements IPerspectiveFactory {

    public void createInitialLayout(IPageLayout layout) {
    }

}
```

说明：

此类必须实现IPerspectiveFactory接口。createInitialLayout是必须实现的方法，在此方法中将配置透视图中需要显示的视图。在本例，现在还只是一个空实现，也就是说MyPerspective这个透视图不包含任何视图。至于在RCP中加入视图的方法和插件开发中的一样，请参考文章后面给出的参考资料。

（2）在plugin.xml的</plugin>前增加如下内容，配置透视图的扩展点，并将此扩展点指向MyPerspective类。

```
<extension
      point="org.eclipse.ui.perspectives">
   <perspective
         class="cn.com.chengang.myrcp.MyPerspective"
         name=" 自定义透视图 "
         id="cn.com.chengang.myrcp.MyPerspective" />
</extension>
```

说明：

- org.eclipse.ui.perspectives是透视图的扩展点
- class—透视图所对应的类（指向MyPerspective类）
- name—透视图的名称
- id—透视图标识。这里把它设置成和class项一样，以后程序中经常要用到此id值，这时就可以用MyPerspective.class.getName()来替代，因为MyPerspective.class.getName()="cn.com.chengang.myrcp.MyPerspective"。这样做的好处就是，不用刻意去记透视图的ID值了。

（3）将MyWorkbenchAdvisor类的getInitialWindowPerspectiveId修改如下：

```
public String getInitialWindowPerspectiveId() {
        return MyPerspective.class.getName();
            //或者  return "cn.com.chengang.myrcp.
MyPerspective";
    }
```

（4）运行此RCP程序，效果如图4。

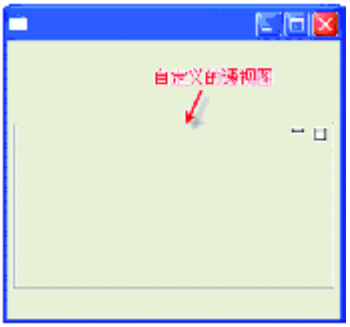


图4：RCP程序加入一个透视图后的效果

配置工作台

工作台（也即主窗体）包含很多UI元素，比如主菜单、工具栏、透视图、状态栏等等。下面将介绍如何配置工作台，具体操作步骤如下：

（1）在MyWorkbenchAdvisor类加入以下两个方法

```
public void initialize(IWorkbenchConfigurer configurer) {
    super.initialize(configurer);
    configurer.setSaveAndRestore(true);//保存界面状态
}

public WorkbenchWindowAdvisor createWorkbenchWindowAdvisor
            (IWorkbenchWindowConfigurer configurer) {
    return new MyWorkbenchWindowAdvisor(configurer);
}
```

说明：

createWorkbenchWindowAdvisor是改写自父类WorkbenchAdvisor中的一个方法，由它指定工作台配置类为MyWorkbenchWindowAdvisor。

（2）创建MyWorkbenchWindowAdvisor类如下：

```
package cn.com.chengang.myrcp;

import org.eclipse.swt.graphics.Point;
import org.eclipse.ui.application.IWorkbenchWindowConfigurer;
import org.eclipse.ui.application.WorkbenchWindowAdvisor;

public class MyWorkbenchWindowAdvisor extends
WorkbenchWindowAdvisor {

    public MyWorkbenchWindowAdvisor(IWorkbenchWindowConfigurer
configurer) {
        super(configurer);
    }

    public void preWindowOpen() {
```

```
        IWorkbenchWindowConfigurer configurer =
                                        getWindowConfigurer();
        configurer.setTitle("我的程序");
        //应用程序的标题栏
        configurer.setInitialSize(new Point(200, 100));
        //初始界面大小
        configurer.setShowStatusLine(true);
        // 状态栏,默认 true
        configurer.setShowProgressIndicator(true);
        // 状态栏上的进度指示器,默认 false
        configurer.setShowPerspectiveBar(true);
        // 选择透视图的工具栏按钮,默认 false
        configurer.setShowMenuBar(true);
        // 主菜单栏,默认 true
        configurer.setShowCoolBar(true);
        // 工具栏,默认 true
    }
}
```

（3）运行 RCP 程序，可见修改后的效果。

加入主菜单、工具栏按钮

通过上面的运行效果，我们发现工作台的主菜单和工具栏按钮一个都没有，这里我们将创建它。具体操作步骤如下：

（1）MyWorkbenchWindowAdvisor 类加入以下方法

```
public ActionBarAdvisor createActionBarAdvisor(IActionBarConfigurer
                                                configurer)
{
        return new MyActionBarAdvisor(configurer);
}
```

说明：

createActionBarAdvisor是改写自父类的一个方法，由它指定工作台主菜单、工具栏按钮的配置类为MyActionBarAdvisor。

（2）创建 MyActionBarAdvisor 类如下

```
package cn.com.chengang.myrcp;

import org.eclipse.jface.action.Action;
import org.eclipse.jface.action.ICoolBarManager;
import org.eclipse.jface.action.IMenuManager;
import org.eclipse.jface.action.IToolBarManager;
import org.eclipse.jface.action.MenuManager;
import org.eclipse.jface.action.ToolBarManager;
import org.eclipse.ui.IWorkbenchActionConstants;
import org.eclipse.ui.IWorkbenchWindow;
import org.eclipse.ui.actions.ActionFactory;
import org.eclipse.ui.actions.ActionFactory.IWorkbenchAction;
import org.eclipse.ui.application.ActionBarAdvisor;
import org.eclipse.ui.application.IActionBarConfigurer;

public class MyActionBarAdvisor extends ActionBarAdvisor {

private IWorkbenchAction quitAction;

public MyActionBarAdvisor(IActionBarConfigurer configurer) {
        super(configurer);
}

protected void makeActions(IWorkbenchWindow window) {
        quitAction = ActionFactory.QUIT.create(window);
        register(quitAction);
}

protected void fillMenuBar(IMenuManager menuBar) {
        IMenuManager fileMenu = new MenuManager("&File",
                        IWorkbenchActionConstants.M_FILE);
        fileMenu.add(quitAction);
        menuBar.add(fileMenu);
}

protected void fillCoolBar(ICoolBarManager coolBar) {
        IToolBarManager toolBar = new ToolBarManager();
        toolBar.add(new MyAction());
        coolBar.add(toolBar);
}

private class MyAction extends Action {
        public MyAction() {
            setText("MyAction");
        }

        public void run() {
            System.out.println("MyAction 被点击了一下");
        }
}

}
```

说明：

- 这个类必须继承ActionBarAdvisor，同时实现父类的一个以IActionBarConfigurer为参数的构造方法。
- makeActions、fillMenuBar、fillCoolBar三个方法都是改写自父类 ActionBarAdvisor。
- makeActions 方法负责生成并注册 action。Action 是 eclipse 一个比较重要的类，可以把它看做是菜单和按钮的动作类。在这里我们创建一个 Eclipse 里内置的 quitAction。而后面的 MyAction 则是我们自己编写的一个 Action。
- fillMenuBar 方法负责配置主菜单，这里是将 quitAction 加入到 File 菜单下，形成一个菜单项。
- fillCoolBar方法负责配置工具栏，这里是将一个自定义的 Action 加入到工具栏里。当然我们也可以将quitAction 加入到工具栏，这正说明了Action的好处：它有很好复用性。
- MyAction是一个内部类，它执行逻辑写在其run方法中。

（3）运行此 RCP 程序，效果如图 5。

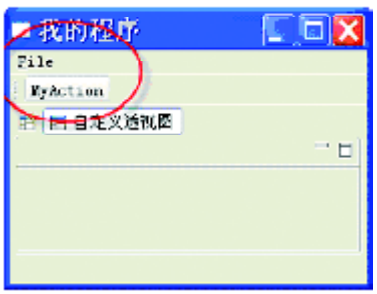


图 5：RCP 程序加入一个菜单和工具栏按钮后的效果

说明：这里工具栏没有显示成一个图标，仅仅是为了简化这个实例，因为它和RCP没有什么关系。实现它非常简单，只需在MyAction类的构造方法里加上其图标设置即可。可参考后面的参考资料。

到这里，RCP所独有的知识内容也就基本讲完了，后面如果要继承改造这个程序则需要更多的SWT/JFace和插件扩展点设置的知识。一般来说，接下来将往透视图里加入几个视图，然再往视图里加入各种界面元素（树、表格、下拉框、按钮等等）。介绍这些内容已经不是本文的任务。

RCP项目的打包发行

上面都是在Eclipse集成环境中运行RCP程序，接下来我们就要把项目从Eclipse环境中脱离出来，打成一个程序包分发给用户。

准备工作

（1）在D盘创建一个目录myrcp，并将原eclipse的startup.jar文件复制过来，再创建一个批处理文件myrcp.bat，仅含一条命令：

```
start javaw -cp startup.jar org.eclipse.core.launcher.Main -application myrcp.MyApplication
```

（2）在D:\myrcp下创建子目录configuration，并将原eclipse的configuration下的config.ini文件复制过来

（3）在D:\myrcp下创建子目录plugins，并将原eclipse的plugins下的以下文件复制过来。

```
org.eclipse.core.commands_3.1.0.jar
org.eclipse.core.expressions_3.1.0.jar
org.eclipse.core.runtime_3.1.0.jar
org.eclipse.help_3.1.0.jar
org.eclipse.jface_3.1.0.jar
org.eclipse.osgi_3.1.0.jar
org.eclipse.swt.win32.win32.x86_3.1.0.jar
org.eclipse.swt_3.1.0.jar
org.eclipse.ui.workbench_3.1.0.jar
org.eclipse.ui_3.1.0.jar
org.eclipse.update.configurator_3.1.0.jar
```

说明：怎么知道需要这些文件呢？参考图3中“Extenal Plug-ins"下所有的勾选项即知。不过需要特别说明的是，org.eclipse.update.configurator_3.1.0在图3中并没有被勾选。

导出项目

（1）选择主菜单“File—>Export…”。

（2）在弹出的Export窗口选择“Deployable plug-ins and fragments”项，然后单击“Next>”按钮。

（3）在Export Destination组选择Directory项，然后定位到“D:\myrcp”目录，同时选择“package plug-ins as individual JAR archives”项。如图6所示。

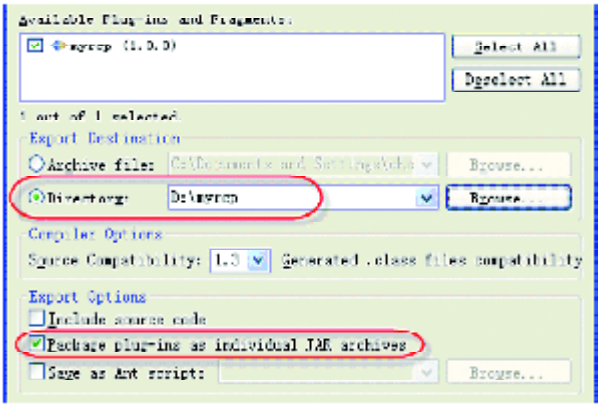


图6：导出项目的设置

（4）单击“Finish”按钮，Eclipse将把项目打一个JAR包（myrcp_1.0.0.jar），并复制到了“D:\myrcp\plugins”目录下。

用户双击myrcp.bat就可以运行这个RCP程序。最后的目录结构如图7所示，myrcp目录下所有文件的总大小是6.26M。

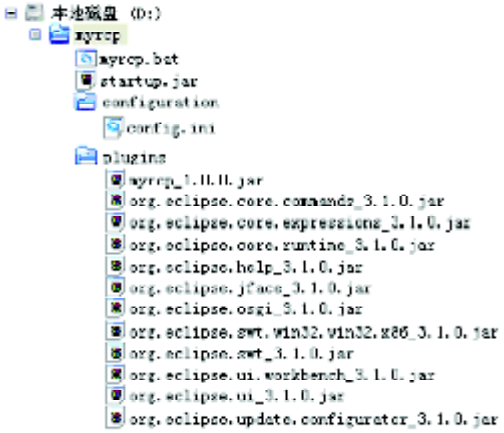


图7：RCP项目的目录结构图

总结

本文主要通过一个步步深入的实例来介绍了一个标准的RCP程序的构建，并介绍了RCP程序的打包发行的方法。当然，这还只是一个开始，本文的知识还不足以开发一个完整实用的RCP程序，你需要了解更多的SWT/JFace和插件扩展点设置方法的知识。但从本实例中，你还是能体验到了RCP的独特之处。

■ 责任编辑：崔轶（cuiyi@csdn.net）

现代软件GUI分析与GUI柔性设计

文 / 孙辉

Win95推出后的十年，软件开发工具日渐成熟了，然而更多的时候，我们看到的总是大公司的产品越来越华丽，而我们的软件依旧看上去很“素”。Microsoft允许你的对象进入Visual Studio .NET的环境，但必须签署一个商业合同，Eclipse虽然是开源架构，但随着其富客户端能力的日渐强大，开发人员的手脚也会被捆绑于其强悍的逻辑框架之内，虽然技术在不断的进步，但技术强悍的公司依旧在寻求它们的游戏规则，使得软件的游戏规则依然因他们而动。

1996年，笔者有幸目睹了还处于测试阶段的Office 97，当鼠标拖动菜单栏那一瞬间来临时，会有一种心为之颤动的感觉，这种感觉会令你激动、难忘。虽然仅仅是个?版，但已经可窥其全貌了，集成的VBA环境足以使你感到震撼，新颖的Form2技术明确的告诉你Office就是一个平台……。直到现在，Office 97依然使得今天的许多软件感到汗颜。随着Office97正式发布，当时的软件界自然的被催生了，其界面规格成了事实上的参照物，各种“97”风格的编程事物有如雨后春笋般纷纷亮相了，Office97造就了Microsoft模仿秀的世界。纵观MS Office的历史，我们不难得出一个结论，MS Office在MS Windows平台上起着一种导向的作用，这种导向首先体现在GUI方面，在Win95推出之前，许多界面特征已经出现在Office应用程序之中，即使现在，我们依然可以看到，新版的Office总是预示着下一代Windows的部分界面特征，许多商业软件的界面特征还在推崇“Office Look and Feel”。

Visual Studio 98的面世，奠定了Microsoft在Windows平台上开发工具的霸主地位。VS98一共包含3个IDE，无论是VC++的IDE、VB6的IDE，还是Visual Interdev 6的IDE，在当时都表现出令人震撼的感染力，直到现在，我们见到过的许多软件，都或多或少的留有Microsoft风格的烙印。通过MSOffice以及Visual Studio，Microsoft为我们展示了GUI世界的美轮美奂，然而不经意间，也将我们的视线引入了歧途，我们经常听人谈起Visual Studio风格的界面、MS Outlook风格界面、MS Office Look and Feel等等，现代界面的感观在一定程度上淡化了软件功能的薄弱，却给我们的软件带来致命的硬伤，我们不得不承认，过度的模仿、表层的似是而非已经使得我们一步一步的陷入了Microsoft等大公司的步调，我们正在深刻的体验形似而神不似的恶果。2002年Microsoft推出了Visual Studio .NET，我们又一次领略到Microsoft的风采，Visual Studio .NET的IDE可谓空前的华丽，Microsoft以多维的幅度向人们展示多窗口驾驭视觉的技巧。通过接触Microsoft Visual Studio Integration Partner计划的详细资料，我们了解到，Visual Studio .NET是一个雄心勃勃的IDE集成方案，理论上讲，在其中可以集成你需要的任何对象，只要你的集成对象是基于Windows平台的。

华丽，永远隶属于豪强？

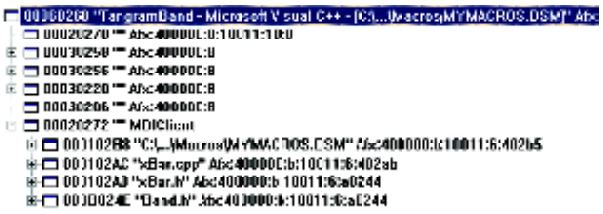
很早以前，大概在MS Word 6.0出现的时候，Microsoft就允许用户直接定制化Word的绝大多数菜单结构、创建新的工具条以及创建可加载项（即插件）等方式定制化应用，由于Microsoft的软件逻辑在底层控制着系统，因此即使是

普通用户的定制化扩展，看上去也很专业，而今天，我们绝大多数的软件开发，即使添加一个按钮，也得进行源代码级别的编程，这一点形成了强烈的反差。

MFC的笨拙给我们留下许多的无奈。1998年，我们曾经为可变尺寸的控制条窗口（来自于VC6的IDE）如痴如狂，每当程序增加一个Docking窗口，我们就多一份欣慰，由此带来的效果是，工程越变越大，心也就感到越发自豪。某一天，我发现，原来VC6的IDE可以自己添加新的Docking窗口（VB6的IDE、VBA IDE以及大多数Microsoft的产品均具备类似的扩展途径），我们感到十分新奇，突然之间我们意识到，我们都错了，Microsoft开发了MFC，但并没有公布MFC应用的秘诀，如同洪七公只传给杨过棒法而不授秘诀一样，我们好像被人带上了一条坦途，但是这是一条平庸者的坦途，我们被纳入了一条别人预定的路径或规则，如果这样，辉煌就将永远属于建立规则的人，我们永远是添砖加瓦的绿叶。

MFC启示录

对软件的内容，MFC类库为我们提供了经典的Doc/View构造，因此，我们被一个毫无韧性的刚性思维模式罩住了，我们用MFC开发的软件，其局限性已经被默认的Wizard锁定，一个10年几乎不变的苍老面孔，使得我们开发的软件永远是具有有限功能的僵化框架。然而，有一个MFC程序，其表现大大超出我们的构造能力，这个软件就是VC6的IDE，如果用Spy++考察VC6，你会确信VC6IDE是个不折不扣的MFC程序：



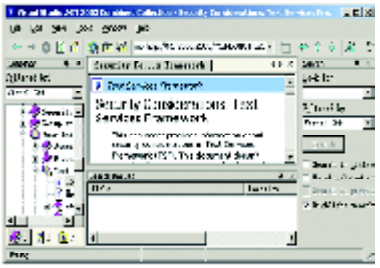
当我们解剖VC6的IDE时，我们发现，同样是运用MFC，Microsoft的MFC应用逻辑与我们的MFC应用逻辑有很大的差别，VC6IDE最初引起人们兴趣的东西是其独特的控制条窗口，这种柔性十足的对象使你感受到一种富有弹性的韵律，充满了Microsoft的风格、味道。也许，柔性GUI框架的感染力就是那个时期爆发的。设计良好、充满柔性的GUI结构，就像一个Hilbert旅馆，只要需要，总会提供满足你个性的拓展空间，在这方面，Microsoft、IBM的许多产品为我们树立了典范。MFC类库是个倍受争议的开发框架，直到现在，MFC仍然是VC++的主要构成部分，在MFC类库的技术构思中，隐含着Microsoft设计风格的深深烙印，笔者坚持认为，既然MFC能够构造柔性十足的VC IDE，那么一定能够造就柔性十足的你，只要你不懈的努力。

MFC有很强的包容能力，因此许许多多用VB、Visual Studio .NET开发的组件均可用于MFC框架，MFC的一个强大的包容能力体现在MFC可以宿主ActiveX控件以及.NET控件，也许是商业利益的原因，虽然Microsoft一直强调MFC在VC++中的重要位置，但MFC的灵活性并没有得以充分的体现。我们注意到，Microsoft在HTML、ASP、以及ASP.NET中，给开发者提供了充分的动态软件特征，而MFC在这方面的表现却不尽人意，不容否认，动态的Web技术造就了今天多彩的Internet世界，然而，1995年以后的10年，桌面软件的表现力与Web界面的表现力在许多方面均形成了强烈的反差，Web界面容易构造、易于管理、表现力丰富、可以充满个性；而相比之下桌面界面则相形见绌，这一切表明，更强大的软件创造力还有待于我们进一步的挖掘。

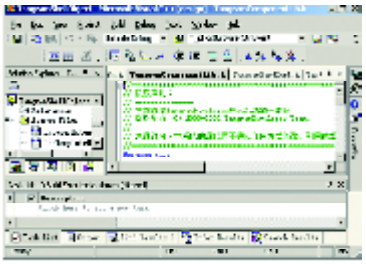
解决问题的路，就在你的脚下

种种迹象表明，许多良好的软件思维，并没有很好的体现在Microsoft等公司提供的开发工具之中，我们开发出的软件，真正是一个一个的软件孤岛，彼此之间缺乏联系、内容贫瘠、刚性十足。而我们的开发之路，却被引上一个偏执的迷途，由于GUI界面的华丽，技术高手竞相模仿，导致软件资金成本、技术成本人为的增加，而软件实质却没有本质的改变，这一点不得不引起我们深思。毒蛇出没的地方，必有克其毒性的花草，这就是所谓的相生相克之理，解决GUI柔性设计问题之关键，也许离你很近。

在我们分析Microsoft的几个产品的框架窗口时，我们似乎发现了一些“近亲”关系，这一点也许会给我们一些启示，我们看看Visual Studio .NET 2003中的两个近亲：



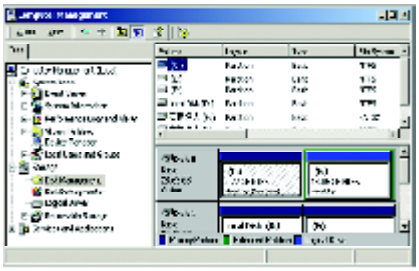
（Microsoft Visual Studio .NET 2003 Document）



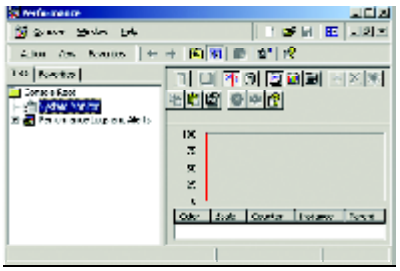
（Microsoft Visual Studio 2003 .NET IDE）

事实上，以上两个框架，完全是同一个程序的两个不同配置产生的实例，类似的界面形式还频繁的表现在

Microsoft的“管理控制台界面（MMC）”之中：



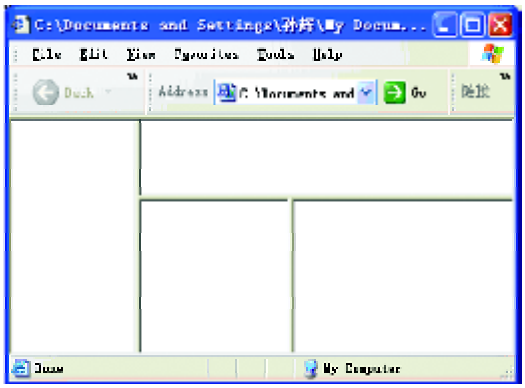
（Microsoft 管理控制台：计算机管理）



（Microsoft 管理控制台：性能）

以上的近亲现象似乎向我们表明，在Microsoft的解决方案中存在着GUI结构的“通解”与“特解”现象，因此，我们需要的可能是一个统一的GUI解决方案，而每个具体的实例，可以解释为这个统一方案的一个特殊配置。

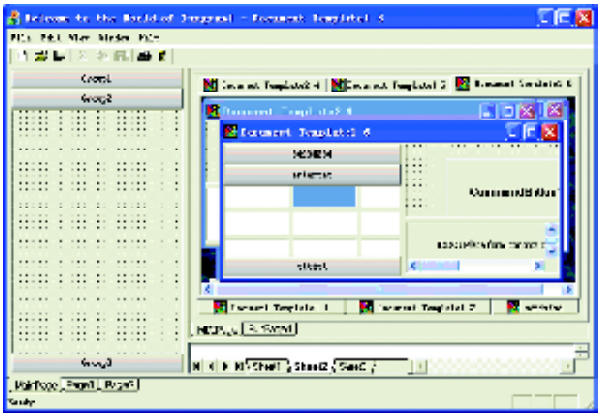
针对一个框架（Frame）窗口，我们已经注意到其四个边中蕴含着特殊的Docking机制构造，这一点，MFC类库已经提供了默认的实现。Frame窗口的另一个功能是可以恰当的维护其客户区，使得客户区内容可以自然覆盖一个最佳的矩形区域。通常，一个客户区可以显示一个或多个用户视图，这一点正是MFC的笨拙之处，用MFC实现一个友好的客户区框架结构通常是很笨拙的，相反，HTML在这方面却表现得极为优秀，用HTML设计一个如下框架：



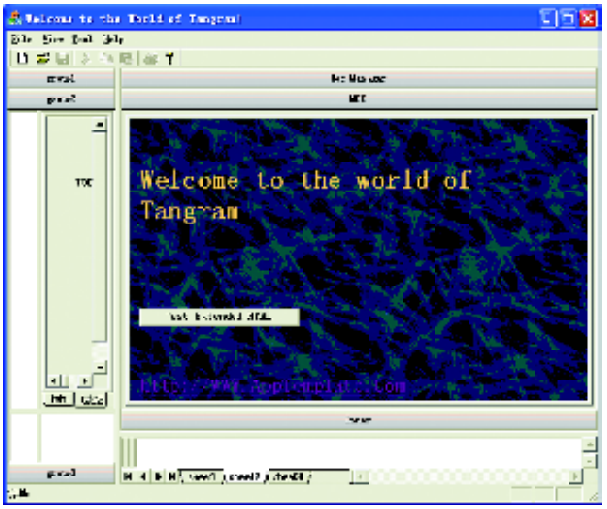
只需一个文本编辑器就够了，相反用MFC实现一个类似的框架结构则不得不借助于C++编程。许多时候，我们会认为HTML编辑与C++界面设计是完全不同的两回事，我们认为正是C++编程中缺乏类似HTML中特有的设计模式，才阻止了C++的广泛应用。C++中繁复、冗余的GUI构造机制已经成为现代软件设计中的顽疾，10年后的今天，这个顽疾依然存在，因此到了彻底铲除这个顽疾的时候了。通过对MFC框架的深入研究，我们发现，许多具有现代特征的GUI设计逻辑，已经被隐含于这个类库之中，因此基于MFC类库的软件框架，天然具备与我们如上分析得到的类似的“近亲”关系，我们考察一下如下所示的两个GUI结构：

显然，这是两个看上去完全不同的框架，因此，按照通常的编程逻辑需要2个完全不同的C++工程分别实现上述框架窗口。对任何从事过C++编程的开发者而言，如上显示的两个MFC框架都不是一件轻松的工作，如果由于某种需要希望对其再次进行界面结构的调整，恐怕都不是程序员愿意面对的事情，Visual C++中，可以称得上“可视化”的部分大概只有“对话框”编辑器，因此，对现代MFC开发者而言，拥有一个MFC风格的可视化的界面编辑器，是十分必要的。

现代程序的一个醒目的特征是窗口的综合运用，在Visual Studio .NET的IDE以及MS Office系列程序中很好的体现了这个特点，功能综合、复杂的窗体的设计明显的成为一个技术问题。从Microsoft的软件中表现出的近亲关系中可以看出，Microsoft可能拥有通用的内部手段一致的构造其GUI成分。基于MFC类库的程序构架，由于缺乏RAD机制，因此对应的技术问题尤为明显。一般而言，绝大多



（一个 MFC MainFrame 框架）



（另一个 MFC MainFrame 框架）

数MFC程序员都将面对下列问题：

1、主窗体的定制化设计,通常情况下,一个程序的主窗体明显的决定了这个程序的轮廓,许多程序的设计技巧视觉上与主窗体的关联很大,而且,不同程序的主窗体通常不同;

2、 文档类型的设计，一个MFC程序通常仅包含有限个文档类型，这一点明显的是一个缺陷，而且MFC的文档设计同样缺乏RAD机制；

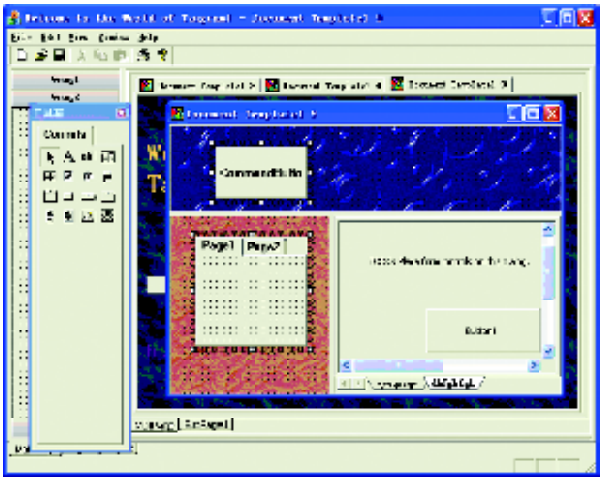
3、 通常情况下，基于MFC的程序仅支持有限个CView的派生对象，这一点不仅极大地限制了程序的表现力，同样也限制了程序功能的内涵、外延；

4、 二次开发机制，缺乏二次开发机制是基于MFC的软件的致命硬伤；

5、 通常情况下默认的MFC程序缺乏灵活、强大的扩展机制，部分基于MFC的程序存在的扩展机制取决于开发者的技巧；

6、 广泛的组件连接机制，这一点是绝大多数MFC程序所不具备的；

7、内置的用户化的可视化设计机制;



（一个基于MFC的软件，图中显示二个Form是基于VB设计的，一个View是CFormView的派生对象）

因此,我们需要一个框架性的工作,使得一个基于MFC类库的程序可以具备以下几个基本特征：

1、一个内置的可视化设计引擎,这个引擎可以设计程序的主窗口，根据不同的需求，可以为系统配置完全不同的个性化主窗口；

2、 允许程序包含任意多个MFC意义下的文档模板,可以以所见即所得的方式设计每一个模板；

3、 允许程序支持任意多个CView派生对象，并保存对应View的数据；

4、 允许程序连接任意ActiveX控件以及.NET Control,允许用户、程序员用CView派生对象、ActiveX控件以及.NET Control设计任意复杂的文档模板；

5、 允许程序具有类似MS Visual Studio Tools for Office一样的二次开发机制；

6、 允许程序自然的集成Microsoft Visual Basic for Application；

7、 使得程序具备类似IBM eclipse模式的Plug in扩展机制。

根据我们的长期实践，我们开发了基于MFC的Tangram软件开发框架已解决MFC开发面对的一系列问题,在这个框架定义了一个被称为“公共组件空间”的一个软件组件集合：我们将ActiveX控件、ActiveX Document、.NET Control以及满足很弱条件的CView派生对象形成的软件组件集合称为“公共组件空间”的“基”，在这个基中取有限个元素（可重复）形成一个有限元素的组，所有这类的有限元素组形成的集合称为“公共组件空间”，由于“公共组件空间”包含的元素是有限维的软件组件组,因此每一个这样的有限维的软件组件组可以对应一个或多个MFC意义下的文档(如果为每个这样的有限维的软件组件组都匹配一个几何布局的话,就形成了一个特定的CFrameWnd)，Tangram的基本功能就是使得“公共组件空间”里的每一个元素自然的成为基于MFC技术的软件的自然扩展，从这个意义上说，每个基于Tangram的应用程序是对“公共组件空间”中元素的数学表示。

Tangram的中文含义就是“七巧板”，是一个以有限驾驭无穷的软件组合构想。这个想法颇具简单、机械唯物主义的特征，我们认为软件就是由大量的简单的“原子”成分组合而成的，所谓软件的GUI视觉差异，可以解释为一种布局差异或结构成分的差异，从根本的抽象层上看，GUI结构的千差万别应该是被剥离在软件编码之外的工作，软件支持的功能单元也是如此。因此软件开发人员需要这样的一类框架，使得软件框架支持的功能不依赖于软件开发人员的代码驾驭能力，GUI形态，应该从开发层自然的迁移到用户层，这是软件开发工具10年变迁、成熟的必然结果。我们认为，就像人们可以根据场合选择服装一样，GUI就是软件的“服装”，应该彻底从软件的设计阶段剥离。

作者介绍：孙辉,现任北京天庄软件有限公司的执行董事兼技术总监，2002年以前在大连海事大学数理系任教，副教授。2002年6月在北京创建北京天庄软件有限公司。1992年开始涉足计算机软件开发，长期从事Visual Studio系列产品的应用开发工作。

■ 责任编辑：崔轶（cuiyi@csdn.net）

用友政务软件性能测试与调优

文/唐仕武(CSIP软件事业部)

软件性能是衡量软件质量优劣的一个重要指标。性能测试是为描述测试对象与性能相关的特征并对其进行评价,而实施和执行的一类测试(例如:描述和评价计时配置文件、执行流、响应时间以及操作的可靠性和限制等特征)。不同类型的性能测试侧重于不同的测试目标,这些性能测试的实施贯穿于整个软件开发生命周期。

性能测试包括负载测试、压力测试和容量测试。其中,负载测试是为了检验系统在给定负载下是否能达到预期性能指标;压力测试是通过不断向被测系统施加"压力",测试系统在压力情况下的性能表现;容量测试针对于数据库,是在数据库中有大量的数据记录的情况下对系统进行的测试。

用友政务软件的特点

用友GMAP平台是针对我国电子政务建设而开发的软件平台,该平台基于J2EE技术、针对政务部门管理应用的政务软件平台。它可以跨平台(Unix、Linux、Windows等)应用、零客户端维护、支持多种大型数据库,其安全保密机制也比较完善(如图1所示)。

此外,用友GMAP平台在技术架构上颇有特色,是一款"平台/部件"技术架构的产品,基于此平台开发的产品由"平台"和"部件"两大部分构成。通过平台可以迅速开发、灵活部署和运行业务系统,而通过部件来实现业务需求。

软件的性能是客户关心的内容,为了提高GMAP平台的和测试平台的性能特性,CSIP结合其在性能测试与调优方面的优势,对GMAP平台进行了性能测试和调优。

性能测试环境及准备

本次性能测试的目标有二:第一,在现有的测试环境下,通过测试发现系统的性能瓶颈所在;第二,在配置的测试环境,能满足系统用户响应时间要求前提下,测试系统承载的最大并发用户数,以及在给定数据量情况下,测试关键业务的响应时间。

测试范围是测试运行GMAP 3.1依赖它的业务模块,通过业务系统的主要功能来反映GMAP平台的性能。

本次测试使用的性能测试工具是Compuware公司的QALoad,它能够在一台机器上模拟多个并发用户对服务器发出请求,同时能够收集系统吞吐量和响应时间等信息。整个测试过程都是在微软Windows Server 2003和SQL Server 2000(SP3)环境下进行测试,因为在系统中保存凭证是比较典型的操作,它对应用服务器和数据库服务器都执行相应的操作,所以我们将以保存凭证的用例为例,对整个测试过程进行说明。

为了更真实地模拟实际应用环境,测试设备是当前主流硬件配置:4×P4 Xeon 2.8G CPU、4G内存、3×146G HDD(Raid 5)、100M网卡;应用服务器的中间件使用了WebLogic Server 8.1(SP4);数据库选择了MS SQL Server 2000(SP3);操作系统是Windows 2003简体中文版。根据业务的需要,我们在数据库中部署了10万条数据。我们测试所用的设备都

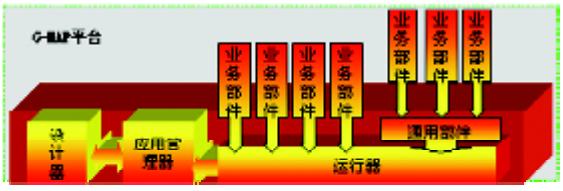


图 1

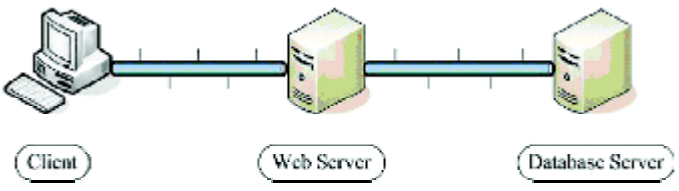


图 2

是当前流行的硬件配置，用三台计算机构成了100M的局域网（如图2所示）。

性能调优过程和测试结果

根据客户的要求，本次性能测试不包括对应用程序的调优，主要是对应用服务器上的中间件以及数据库进行优化。我们调优的方法是对系统进行加压，直到系统出现错误或服务器的CPU利用率持续超过75%，找到系统的瓶颈，再通过对影响瓶颈的因素进行优化。

首先使用测试工具QALoad运行用例保存凭证，我们开始运行10个并发用户，之后每2分钟增加10个并发用户的方式来对系统进行加压。系统在120、140和170个并发用户数时，应用服务器的CPU利用率也超过75%，但是数据库服务器的CPU利用率只有18%左右，具体参数如表1所示：

表 1

并发用户数	APP 的 CPU 利用率	DB 的 CPU 利用率
120	78%	16%
140	77%	15%
170	78%	18%

当并发用户数大于120时，系统出现异常，通过监视中间件WebLogic的日志，出现以下信息：在处理系统异常的过程中发现response已经提交或getOutPutStream方法已被调用。系统异常为Java.Lang.Illegal State Exception——定向到/ListPage发生IOException，Java.net.Socket Exception——Software caused connection abort，Socket write error。

根据以上测试的结果，经过专家组的分析与用友技术人员的讨论，我们认为系统的瓶颈是在应用服务器上。由于本次测试不包括对应用程序的调优，所以只对中间件WebLogic进行优化。

我们请BEA公司的技术专家来到现场进行分析和讨论，讨论结果是对中间件WebLogic以下方面进行优化：比较不同的Java虚拟机、比较不同的参数Java虚拟内存、比较不同的Java执行线程计数、比较不同的JDBC连接缓冲池中初始容量。因为这些组件和参数是影响系统异常和应用服务器繁忙的因素。

由于客户能够容忍的响应时间为8S，我们在用例保存凭证上运行20个并发用户，比较系统在不同公司的虚拟机J2SDK 1.4.2.08和Jrockit 1.4.2.05、以及不同Java虚拟内存1024M和1536M、不同Java执行线程计数和不同JDBC连接缓冲池初始容量的测试结果，测试结果如表2所示。

比较以上的测试结果，考虑到该测试环境的内存为4G，为了保证系统在保存大数据量的时候内存不发生溢出，我们认为该系统采用BEA公司的Jrockit 1.4.2.0虚拟机、设置Java虚拟内存为1536M、Java执行线程计数为40以及JDBC连接缓冲池初始容量为40，该配置是我们理想的系统配置。

客户一般能够容忍系统的响应时间为8S，测试系统在上面比较理想的系统配置下的最大并发用户数为（见表3）：

表 3

并发用户数	响应时间（S）	APP 的 CPU 利用率（%）	DB 的 CPU 利用率（%）
22	8.012	8	21

总结

经过CSIP的性能测试和调优，GMAP平台的性能得到了较大的提高，提高了用友政务软件的质量，使用友政务的产品在市场上有了更强的竞争力。从测试结果看，该软件的性能表现地很稳定，在微软平台上部署一个200—300用户的运行环境是完全可以承受的。

注：信息产业部软件与集成电路促进中心（CSIP）是信息产业部领导建设的，旨在引导产业发展、为我国软件与集成电路产业和企业提供公共服务的中立、开放的机构。CSIP拥有一批具有丰富测试经验的技术专家，能为各企业提供功能测试、性能测试与调优、黑盒测试、白盒测试等各种测试服务。CSIP以中立、开放的立场，为企业提供各种测试服务，帮助企业解决各种难题，提高企业软件的质量，促进企业提升核心竞争力，推动在国内企业之间、中外企业之间形成深层次的合作，在软件领域中促成良性产业链的形成，推动我国信息产业的跨越式发展。

表 2

Java 虚拟机	Java 虚拟内存	线程计数	JDBC 连接缓冲池初始容量	响应时间（S）	APP 的 CPU 利用率（%）	DB 的 CPU 利用率（%）
J2SDK 1.4.2.08	1024M	40	35	9.045	31	13
Jrockit 1.4.2.05	1024M	40	35	7.263	7	18
Jrockit 1.4.2.05	1536M	40	35	7.426	7	18
Jrockit 1.4.2.05	1536M	40	40	7.318	7	19
Jrockit 1.4.2.05	1536M	60	40	7.73	7	19
Jrockit 1.4.2.05	1536M	120	40	7.328	7	20

JBuilder 2005的百科全书

文/黄甫

在目前流行的Java开发工具中，JBuilder 2005仍然是功能最全面、最强大的一个。我有一个朋友参加了Borland JBuilder的认证培训之后说，这款开发工具已经做到了一般开发团队所能想到和用到的全部。可惜的是，由于中文文档资料的缺乏，很少有人能够一览JBuilder的全貌。电子工业出版社博文视点公司最近出版的《精通JBuilder 2005》填补了这一空白。这本书不但全面深入地介绍了JBuilder这个工具本身，而且包含了很多技术知识和经验，不但对于JBuilder用户有着很好的参考价值，对于那些希望尽快掌握Java高级开发技术的初学者也是一本极好的参考书。不但如此，这本书来自开发一线经验丰富的实践型专家，整体质量非常高，是目前本类图书市场上一个高质量的典范。

这本书最大的特点是全面细致、重点突出。目前的Java软件开发技术包括的内容已经十分广泛了，任何一本书都不可能、也不必要包罗万象。但是对于学习者和开发者来说，如果在头脑中没有一个全景，那么对于技术的认识就必然片面，选择学习线路或者面对具体开发任务的时候就难免盲目，造成时间和精力的浪费。Borland在设计开发JBuilder的时候，已经根据自己的经验选择了把Java技术中最重要、最活跃的部分融入到JBuilder工具中。因此，具体的学习一个工具，反而成了目前迅速把握Java技术主干的一个捷径，但是这一定需要一本全面而又重点突出的书加以指导，本书在这方面表现非常突出。全书共分成五个部分，第一和第五部分专注于JBuilder工具本身的使用，而第二、三、四部分则专注于对JBuilder所支持的主要Java技术进行全面而扼要的介绍，内容既涉及到JavaBean组件开发、Web开发、Servlet/JSP、JSF、Struts、XML、CORBA、Web Services等标准主流技术，也涉及到JBuilder特有的可视化开发、JDataStore数据库、DataExpress数据库访问技术，非常全面。一个初步涉猎Java的读者，完全可以以这本书为导航，将目前主流的技术进行一个完整的梳理，从而做到胸中有大局。而在工作中继续解决实际问题的读者，也一定可以从本书中找到启发。对于那些希望学习诸如Struts、JSP、Web Services等流行技术的读者来说，使用JBuilder这样的先进工具，配合这本书的讲解来学习，可以起到事半功倍的效果。



《精通JBuilder 2005》

本书的另一个特点是实用。Borland的产品有一个通病，就是文档资料不够丰富。做为目前Borland开发工具序列中最重要的产品，JBuilder一直受到优待。学习者不仅可以从网上下载一套内容丰富的电子书，而且Borland官方的技术文档也被翻译成中文并且发行于市。既然如此，本书的意义何在呢？相信读过这本书的读者都能得出结论，那就是它的实用性。现在市面上有不少技术图书，不是来自产品技术文档，就是由"职业写手"根据技术文档改写而成，内容浅薄，形式造作。而这本书来自具有丰富经验的一线开发者，随便阅读几个章节都可以感觉到作者的实践经验，对于问题的选择和解释，背后都有实在的理由。通过这本书来学习实践技能，能少走不少弯路。

去年以来，逐渐出现一批由中国本土开发人员撰写的中、高端技术图书，由于他们经验丰富、认识深刻，而且具有表达上的优势，对中国实际产业状况更为了解，所以写出来的技术图书更接近国情实际，更容易为读者所接受，受到了读者的欢迎。本书就是典型的本土技术图书精品，我们乐于向广大Java学习者和一线程序员推荐。

现实中的软件工程

文 / Aimingoo

编者按:很多执着的技术人员对软件工程不屑一顾,这其中的误会更多地源自于他们对于“工程”域模糊或偏激的理解。更多的情况是:一人一把号,各吹各的调。如果不能够领悟工程之要义,那就将难于协调团队中的角色,也难于自发地成为能被协调的团队角色。一旦你希望在项目的“工程”中得到更宽广的视野,从根源中去理解“工程”的价值、对立与决策,那么Aimingoo的这本小册子一定会让您从另外一个角度看待您眼下的工作。本文节选自《从编程到工程》一书的第八章。

“王不如远交而近攻,得寸,则王之寸;得尺,亦王之尺也。”

——《战国策.秦策》

大公司手中的算盘

从最早仅仅关注于软件开发工具到现在,软件行业中的巨头们已经在层出不穷的思想中涅槃了一回又一回。

Rational被IBM并购的真实原因在于IBM需要构建一个完整的软件工程体系,有了Rational的IBM会变成这个样子(见表1)。

IBM得到Rational的最大好处是在软件工程方面,快速地拥有了一套成熟的理论体系和实作工具。对于IBM来说,Rational有着UML语言非常丰富的实践经验,还有着RUP作为理论框架的创立者和领导者的地位,这些对IBM在确立大型软件工程应用方案提供商的行业形象上都是极大的支持。

在语言方面,IBM注意到Java作为平台中立的语言特性,以及它在大型应用工程方面的成功表现。作为扼制Microsoft的平台优势的唯一途径,IBM在语言方面选择支持Java是明智的。

出于同样的理由,IBM亲近开源软件界,并很快得到开源软件领域的头羊地位。这使得IBM从没有语言优势立即变成了“可以忽略语言劣势”。开源界给了IBM一种对抗的背景和实力,而IBM只需要做到把握这种力量,就可以在潮流中稳如磐石。

把握力量总之比创造力量来得经济。

同样,Borland也从开发工具厂商的位置中跳出来,希望构建类似的软件工程体系,所以现在你会看到这样的一个Borland(见表2)。

对于Borland来说,对软件开发语言(C、Java、Delphi)把握是其自身优势,所以Borland一直保持在语言上的中立,以寻求一种在不同平台上的开发者社群的支持最大化。Borland积极的推动UML的标准化,一方面可以使得Borland有机会在模型语言标准的制定上有机会制造影响,另一方面也可以快速地与IBM/Rational构成对抗Mircosoft的战线。

作为工具开发商,Borland快速地拥有了实现ALM(Application Lifecycle Management)所需的绝大多数软件产品。然而Borland也很快意识到,(当前的)ALM是一个产品体系而不是一个理论体系:Borland没有在ALM作为工程理论方面上的任何优势。于是Borland开始并购与实现ALM体系相关的公司,其中收购过程改进咨询公司TeraQuest并组建流程优化实务部,以及收购TogetherSoft为开发工具增强模型构建能力,都是相当大的一些举措。通过这些努力,Borland快速地补全了ALM作为一个工程体系在理论方

表 1

IBM软件工程(2004)		
	理论体系	实　　现
工具	Language	Rational Suite、WebSphere、Eclipse
方法	OOA/D/P	IBM 软件开发平台(SDP)
过程	RUP	RUP2000、RUP2003

表 2

Borland软件工程(2004)		
	理论体系	实　　现
工具	Language	Togther、StarTeam、Delphi、CBX、JB...
方法	OOA/D/P	Galileo、PrimeTime
过程	ALM	Borland ALM Solution

面的不足。

对于IBM来说，RUP和UML是优势，所以IBM用来削弱Borland在开发语言上的优势的最佳手段，就是支持开源的Eclipse，以及用UML的标准化来确立其规范制定者的地位。然而你会惊异的发现，Borland一方面在支持UML的标准化，另一方面还在支持着Eclipse的开发并协助其快速成为一个完整的、具有商业品质的开发平台。这似乎是极其怪异的战略：帮对手磨剑。

如果Borland只为一个对手磨剑，那他可能是一个傻子。但问题是，Borland几乎为他所有既已成为(或者终将成为)对手的人磨剑：Kylix是Linux平台上的产品；C++ Builder、C# Builder、CBX、Delphi是Win 32和.NET平台上的产品；JBuilder则是Sun平台上的产品。—— 一切正如Borland自己说的那样，他是“(语言、平台和技术)中立的软件厂商”。

Borland走在钢丝绳的中间，对他的考验是平衡的艺术和技术：如果他倒下，钢丝绳两端的任一方都来不及施以援手；然而如果他存在，那么他向哪边迈出的一步，就给对方以最大的压力。

“敌人的敌人就是自己的朋友”，聪明的战略家总是能看到这一点。然而Borland却力图使这个敌我都分不清的战场呈现出一种古怪的格局：一方面Microsoft是Borland的股东之一，另一方面Borland在做Sun、IBM以及Linux平台上的软件提供商。

与Borland和IBM通过通过并购来达到目的方式，Microsoft有足够的力量全方位出击，因此你看到的体系如表3所示。

表 3

Microsoft软件工程(2004)		
	理论体系	实　　现
工具	Language	VS.NET、DSL、.NET Framework
方法	OOA/D/P	需求方法、模型方法、测试方法...
过程	MSF	MSF Process Model v.3.1

Microsoft在工具、方法和过程方面都有具体的实现。而IBM在方法和过程层面上大都停留在理论阶段，Borland在这些方面上虽有丰富的产品实现，却又相对缺乏理论基础。

Microsoft并不仅停留于此。从.NET Framework提出开始，Microsoft就试图在开发语言和基础框架上实现大统一，希望能达到UML在模型语言中的地位。因此出现了CLR+CTS通用语言体系以及其具体的实现：.NET CLR+IAsm。.NET上的代码要求最终被实现成中间代码，可以反汇编到IAsm，这意味着任何其它公司在开发语言层面上的优势丧失殆尽，所以开发者们看到C#、Jscript .NET和VB .NET同期实现的“壮举”。

而Mono的出现，对于Microsoft而言是绝对的福音。Microsoft把.NET Framework中的C#、公共语言架构(CLI)以及通用类型系统(CTS)等做成ECMA标准，最期望看到的就是类似Mono这样的第三方产品的出现。事实上，Mono做了Microsoft从来都想做而不敢做的事——解决了Microsoft产品的跨平台问题，进而削弱了Sun相关语言的跨平台优势。Microsoft一方面不想放弃自己的平台优势，另一方面又为Sun的跨平台优势所制肘。而Mono的出现以及它适度的影响力，正好成为Microsoft平衡这种微妙的、相对优劣形势的棋子。

接下来Microsoft开始向模型语言发难。领域专用语言（Domain-Specific Language，DSL）的提出绝非偶然，那是在硝烟未尽的战场上重新燃起的战火。

软件业界如今的局面，不是一些人(例如程序员或者评论家们)争争吵吵的结果，而是大公司们相互制衡的结果。Borland与IBM、IBM与Sun以及Sun与Apple都在做着相同的事，又都有各自的算盘。他们一边压制对手的优势，一边又借助对手和同盟的力量来削弱自己的劣势或者补充实力。

跳到局外来看，并不是说Microsoft是他们的共同对手，而只是因为Microsoft占在了峰头浪尖，便成了众矢之的。所有人面对的并不是Microsoft的名字，而只是它的地位，无论谁成就了这个地位，都将承受相同的风险与压力——当然也包括机会。

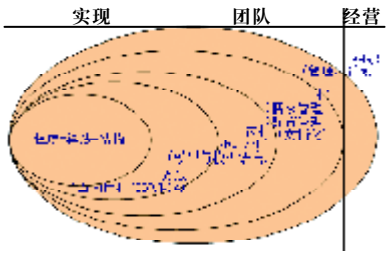
众多大公司在标准、理论、语言上的争来夺去，未必全部出于“软件实现”的考虑。对统一理论、统一工具、统一过程的企图，其最终目的是在整个软件工程体系中的全面胜出。

算盘上的绝大多数人，只是用于计算胜负的一枚算子。

回到工程的关键点

除了软件本质力量的推动之外，商业因素也推动着软件工程体系的发展。大公司之间的争夺战的最终结果，已经开始把软件工程从原始的“自生演进”状态，逐渐推进到“他激发展”的状态上了。

这种他激发展可能会影响到软件工程发展的速度，然而在各个工程层面上的关注点并不会发生变化。在前面的模型图中，每一条纵向的细线用于定义一个关注点[1]。我在另一次培训中为这些关注点加上了标注：



图：软件工程层状模型

上图标示的模型被我命名为软件工程层状模型(EHM，Engineering Hiberarchy Model)。与“牛屎图”所代表的“软件工程体系层次”不一样的是,EHM不描述工程元素间的关系,甚至在试图割裂这些元素以使得工程角色定位以及各自的视角更加清晰明确。

从这个模型中可以看到,在“程序”与“方法”层面,是关注于“(具体的)实现”的;而在“过程”和“工程”层面,首要考虑的是团队问题。从角色的角度上来说:开发经理思考项目的实施方案和管理具体的开发行为;而项目经理则保障团队的稳定性和一致性。

然而这只是基本模式,或者说,是理想模式。

思考项目成本的经理

在标注关注点时,如下的问题引起了我的思考:

- 项目的管理到底是组织管理还是成本管理?
- 项目的计划到底是组织规划还是成本计划?

一言以蔽之:项目管理要不要考虑成本问题?

现在,从一个细节跳出来,进而分析我们的角色。这个细节就是:如何完成今天的工作。

正如前面所说,如果你是一个软件公司里的项目经理,你今天的工作可能是写一份项目计划案或者听测试部的报告,亦或安排会议来听取和分析一个新的产品需求。然而,我需要说的是:这是细节。

细节就是你使用的Project 2003,或者你正在公司内部署和推广的ClearCase。如果它们正好是你今天要完成的工作或者是你明天要用来工作的工具,那么,做为项目经理的你,现在就要立即跳出来。

理想状况下,“软件工程=过程+方法+工具”。然而,工程成功的真正关键并不在于你把你的团队“组织”得有多好。即使在团队中他们都显示出有条不紊,你一样会面临失败。

蚂蚁的团队总是被本能地组织得非常好。然而如果一个蚂蚁的群体中有了流行疾病,致使蚂蚁死去,而新生蚂蚁不能跟上其死亡的速度,那么这个团队很快就溃散了。

这是因为蚂蚁用于维护团队运作的“资本”在流失。如果资本没有了,就没了运作,团队的存在就没有了必要性和可能性。项目就死亡了。

埋头于画甘特图的项目经理犯下了与挖山不止的愚公同样的错误:忽略了成本。

如果愚公真的可以成功,那么可能是300年之后。然而如果一个工程要300年才能做成,那么在做成之前,客户就选择了放弃。

如果有机会,项目经理可以选择向另一家公司购买一个产品来卖给客户,从“为客户开发”变成“为客户定制”以及“为客户服务”,这样就在没有任何开发成本的前提下完成了工程。与另一个同样极端的例子相比,你会发现它与前面那个“做过场”的项目全然不同。后者是做完了工程,却没有做成工程。而现在这个项目经理却做成了工程,但是在许多的过程环节上他根本就没有开始。

然而,现在除了跃跃欲试的技术经理之外,没有人会不满意这个结果。

技术经理最常说的话是:我们可以开发出来;开发人员最常说的话是:我可以开发出来;愚公最常说的话是:何苦而不平?

还记得那句话吗——不要栽进蚂蚁洞里!

愚公如果停下来,思考的问题可能是碎石的“方法”。而项目经理从细节中跳出来,思考的问题就应当是完成工程的“方法”。评价这个方法的好坏的标准只有一个:节约成本。

Y公司由K公司过渡而来的时候带来了一个市场前景非常看好的产品。而存在的问题则是两方面的,一是扩大市场占有率,二是继续技术投入。

于是,Y公司请来了专家D。他是一个在行业中摸爬滚打了多年的顾问型专家:做过公司,也在无数个公司做过。D先生的项目计划可能是无可挑剔的,但其投资规模决定了它无法实施;D先生在一些产品计划上的思考上也是切近市场的,然而他没有学会如何为团队争取到两名以上的开发人员;D先生在部门管理上的方法也是适当的,然而他忘记了训练部门人员以使他们与自己保持一致的步调和方向(组织和管理一个松散的团队比照顾一群蚂蚁难得多)。

于是在Y公司建立到倒掉的四年时间里,D先生三进三出,营销计划一再被否决,而产品的再研发计划也数度被搁置。很快,这个并不生动的故事终结于我跟他的最后一次会谈:三年之后,产品彻底从市场中退出。

“思考成本”,这是D先生给我的教训:

- 不计成本的项目计划不会得到经营者的支持;
- 毫无目的地消耗成本是项目中的慢性毒药;
- 最致命的风险是成本的枯竭[2]。

①我画出的的确是线而不是点,“关注点”只是一个概念。如果你非要去发现一个“点”,那么你可以用几何的目光,关注于弧线与直线的切点。然而,这样的结果将是你彻底的忽视了“关注点”的本质含义。

②我经常注意到的成本因素包括时间、人力、资金和客户成本。而大多数情况下,人们不会把客户的数量以及耐心当做(客户)成本来计算。而在我的项目规划中,这是成本。

Amazon.com开发技术图书每月观察（2005年7月）

文/刘铁峰

本月的Amazon技术类图书榜单的总体变化不大：Microsoft Press的《Code Complete》再次雄居榜首，O'Reilly的《Head First Design Pattern》尾随其后。这两本书可谓不分伯仲，更有意思的是，近四个月来，每个月它们的位置都交换了一次，却始终排在前两名。相对于具体编程语言类技术书籍，这种对普遍程序员适用的书籍有着更稳定的读者群，并且一旦成为经典，则长销不衰。出版于1995年的《Design Pattern》仍位于榜上第十名，在国内各大计算机图书销售网站也排名前列。位于榜首的《Code Complete》是2004年才出版的第二版，其第一版则成书于1993年。按照现在的趋势，恐怕还会一直地畅销下去。值得令人期待的是，《Code Complete》第二版已经由博文视点公司引进，在年内应该能够看到这本经典书籍的中文版。

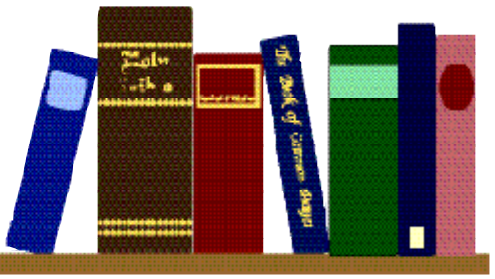
近两个月以来一直排第三位的《Hibernate In Action》一书小降一位，排名第四。其势头依然强劲，这也充分证明了Manning公司的“In Action系列”深得程序员的喜爱。排名第五的书籍，虽然在预料之中，但其上升的速度实在令人吃惊，因为它上月在榜单上才排第69名——这就是让大家期待以久的、“Spring之父”Rod Johnson所著的《Professional Java Development with the Spring Framework》。和Rod Johnson一贯的风格一样，这本书并不是介绍如何使用Spring，更多的是强调Spring能够做什么，为什么要用它以及在什么时候要用它。相信和作者的前两本几乎改变了Java世界的书籍一样，这本书也会保持作者的一贯水准。Rod的一度在5月份冲进前十的《Expert One-on-One J2EE Development without EJB》现已降至第35名，而同样在5月份位居14名的《Expert One-on-One J2EE Design and Development》已经离开了排行榜。虽然Rod Johnson的新书得到了程序员的热捧，但是Spring的读者群毕竟有限，关于Spring的两本书籍《Spring in Action》和《Pro Spring》的排名也分别降至36名以及90名。虽然不知道Rod的这本新书能够在排行榜上停留多久，但鉴于其号召力，或许会更加推动Spring的发展并带动其他Spring图书的销售。

在整个排行榜中，O'Reilly的“Head First系列”非常打眼。O'Reilly的图书质量一直深得程序员信赖，在本期的排行榜中，除了一直遥遥领先的《Head First Design Pattern》之外，《Head First Java》（2nd Edition）和《Head First Servlets & JSP》分别居于第九、第十二名。由此可以看出Java系列的书籍仍然受到了更多程序员的关注。

Amazon软件开发技术图书精选排行榜（2005年7月15日）

名次	上榜图书	作者	出版商
1	《Code Complete》（Second Edition）	Steve McConnell	Microsoft Press
2	《Head First Design Patterns》	Elisabeth Freeman	O'Reilly
4	《Hibernate in Action》	Christian Bauer、Gavin King	Manning
5	《Professional Java Development with the Spring Framework》	Rod Johnson, et al	Wrox
9	《Head First Servlets & JSP》	Bryan Basham、Kathy Sierra、Bert Bates	O'Reilly
10	《Design Patterns》	Erich Gamma、Richard Helm、Ralph Johnson、John Vlissides	Addison-Wesley Professional
12	《Head First Servlets & JSP》	Bryan Basham	O'Reilly
21	《Expert .NET Delivery Using NAnt and CruiseControl.NET》	Marc Holmes	Apress
35	《Expert One-on-One J2EE Development without EJB》	Rod Johnson,、	Juergen Hoeller Wrox

新书上架



CSDN每月精选上市的新书,由编辑和业界专家亲自点评。诚邀读者推荐新书!

《C# 高级编程》(第3版)

出版社:清华大学出版社

C# 是为在 Windows 平台上配合.NET完成开发工作而设计的一种编程语言。开发人员在利用C#进行开发工作的同时,还可以了解和认识.NET架构的工作模式和背景环境。在第三版《C#高级编程》中,读者不仅可以学习到C#基础知识,而且还可以深入了解应用程序的开发和部署。读者通过这本书不仅可以获得C#本身语言的知识,而且还可以利用这种语言在.NET环境下创建Windows窗体、利用ADO.NET访问数据库、为ASP.NET编写组件,以及进行相关的COM和COM+开发工作,等等。第三版《C#高级编程》是完全为开发人员"量身定做"的,它包含了针对于.NET Framework 1.1和Visual Studio .NET 2003的执行代码和相关实例。

《Eclipse从入门到精通》

出版社:清华大学出版社

Eclipse是目前势头正盛的一种IDE,它在最初由IBM和OTI的IDE产品开发组于1999年4月开始创建。Eclipse并不是单纯的Java编译器,通过使用插件可以编译诸如JSP、C++等多种语言,本书的作者本人就具有Eclipse SWT/JFace插件开发经验。《Eclipse从入门到精通》的读者面向群是使用AWT/JFace开发C/S客户端程序的开发人员和使用J2EE开发B/S Web应用程序的开发人员。本书开始以五章的内容做为"起步篇",介绍了Eclipse的安装、集成开发环境以及在Eclipse中CVS的使用。作者对"SWT/JFace篇"和"插件开发篇"重施笔墨。在SWT/JFace篇,主要针对于Java客户端的开发人员,讲述了SWT/JFace中各组件的使用以及相关类的介绍;插件开发篇中,作者以实例详细介绍了开发一个完整Eclipse插件项目所需的具体知识和开发流程。其实,读者可以抽出身来看一下,可以把这两篇和为一篇,即"SWT/JFace插件开发篇"。作者对SWT/JFace和插件投入如此的笔墨与精力,可谓"干什么吆喝什么"。本书的最后,以四章的内容介绍了Eclipse的Web开发:Eclipse中Web开发环境的搭建、Struts的使用、Hibernate的使用。



《敏捷项目管理》

出版社:清华大学出版社

敏捷软件开发从它一出生开始便饱受争议。然而,在工程领域中重量级和轻量级的软件工程方法,归结起来都是源于实践的积累和认识。人们通过长期的实际工作,总结和归纳出系统的实施方法。这一点和人们通过上学——从书本中获取知识的方式截然相反。因此,软件工程类的图书很多,优秀者却寥寥无几。所幸本书邀请了两位具有丰富经验的项目经理,他们通过不同文化背景介绍其在软件项目实施中的经验。因此,本书的第2章和第3章近四十页的内容,完全能够值得这本书的书价。

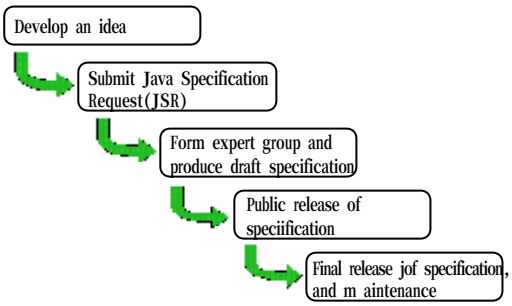
中国企业走近 JCP

文/黄海波

对Java开发人员，JCP（Java Community Process）这个名词并不陌生。但对国内大部分Java开发人员来说，JCP更多的是一个符号，一个国际Java开发社区的象征。而对JCP这个组织的来源、组成、运作模式以及JCP对中国软件产业，甚至是我们自身工作事业的影响可能不甚了了。由于历史原因、文化语言的差异，国内Java厂商一直未能对JCP引起足够重视，从而导致国内的软件厂商无法参与到JCP的行业标准的制定过程中去，结果就使我们只能跟随制定好的标准，而不能影响标准向着有利于国内软件产业的方向发展。

跟随的劣势是很明显的，以万众期待的下一代Java持久化标准EJB 3为例。EJB 3规范目前仍在早期规范阶段，预计要到２００６年中期才能完成最终版本，但EJB3 专家组中的Java厂商都已经根据讨论的初步意见开始了产品开发，有些甚至开始发布预览版本。而国内的J2EE厂商却可能要等到EJB3的最终版才可以着手进行研究和开发（根据早期规范不可靠，变动通常很大），差距自然巨大。在其他诸如如商业智能、工作流、数据挖掘等领域都存在类似情况。

可喜的是国内软件业的Java行业组织和管理部门都已经认识到，继续徘徊在JCP的门外不利于国内Java产业的生存和发展，也和国内庞大的软件市场不相符合。在许许多多有志之士（很多是笔者的前辈，他们为国内软件业发展默默努力，实在让人钦佩）的共同努力下，JCP的中国分支也即将成立。而作为第一个尝试，中国的Java应用体验认证实验室在国家应用软件产品质量监督检验中心、Sun计算机系统（中国）有限公司、中国软件行业协会中间件分会、北京软件与信息服务业促进中心四单位共同努力下在北京成立，为中国的Java软件厂商在国内进行软件的JCP认证和兼容测试。



图：JCP 流程

表：JCP 纵览

Step	Timeframe
Participant submits a new specification, or JSR	(Anytime)
Other participants review the JSR	7 to 30 days
PMO reviews comments	(Accepts, rejects, or defers JSR)
Call For Experts (CAFE)	At least 15 days
PMO selects specification lead	(Most capable person chosen from CAFE nominations)
Specification lead forms Expert Group	(From other CAFE nominations and other sources)
Expert Group writes draft	However long it takes
Participants review draft	At least 30 days
Specification lead revises draft	Up to 30 days
Public reviews draft	At least 30 days
Specification lead revises draft	Up to 30 days
Public release	Depends on completion of CTS and RI
Expert Group develops CTS and RI	(Specification lead is responsible for completion)
Expert Group decides that specification, CTS, and RI are complete	However long it takes
Final release of specification	(Everything complete)
Maintenance of specification	(Specification lead assumes responsibility for maintenance)

构造随需应变的企业级软件
EOS™面向构件的中间件
www.primeton.com



打造面向构件的大型企业级应用

文/黄柳青

今天给你一粒米，明天给你两粒米，后天给你四粒米，如此这般，一个月后你有多少米？想必大家都很熟悉这个入门级的游戏，软件也是这样。随着需求的增长、功能的变化、工期的延长，在你抬头一望之间，我们在课堂里能编写出来的小白兔一下子变成了异常可怕的怪兽：它有几百万个关节，每个都能自由转动360度！实际上，大型软件和小软件根本就是不同的物种，需要区别对待。最具挑战性的就是在开发大型软件时，需要在需求多变的不可测性中和在容量、交易量的不断突破中，来打造一个长期稳定的应用架构。在如此环境中，我们的应用架构要同时支持多方面相对独立的成长，即数据结构的成长、系统容量和性能要求的成长以及业务逻辑的成长（参见图1），从而打造一个可以成长的大型企业级应用软件。

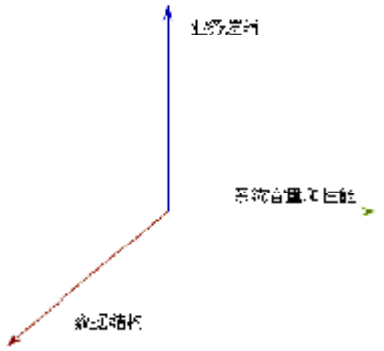


图1：应用架构应同时支持多方面相对独立的成长

数据结构的成长

我们曾经讨论过如何使用标签化数据来解开业务逻辑和数据结构之间的锁链。XML是标签数据的典型案例，不过标签数据并不一定要时刻通过XML一类的字符来表达。一般来讲，数据在跨系统的交互时才会通过XML来表达。通过标签化的参数传递，我们可以把数据结构与业务逻辑分离最大化，让业务逻辑自动适应数据结构在一定程度上的成长和变化。

业务逻辑的成长

随着软件的应用越来越深入、系统越来越大、功能点越来越多。如此复杂的系统，必须由掌握不同技能的人来共同完成。软件成了名副其实的“系统工程”，人员数量也成了软件复杂性的一个参数。现在，“On Demand”这个词在软件业界已相当流行，那么Demander又是谁呢？是Business！软件的变化是瞬息万变的商业环境决定的，因为软件系统越来越成为商业运作的内部引擎；提供不出功能强大的、安全可靠的、支持分布业务和高数据量要求的软件系统，就只有“客户很生气，后果很严重”这个结果了。

确实，软件必须成长，业务逻辑必须成长，企业需要“活”的软件。

面向构件：否定之否定

纵观大型软件的系统架构的变化，竟能发现哲学！

从原始的一锅粥架构到垂直分割的结构化架构，是第一次“否定”。现在看来，根据功能把软件的“解决方案空间”进行“竖割”虽然耦合度较高，但这个缺点不能掩盖产生“竖割”的重要原因——这其实是从用户视角对软件的“问题领域空间”分割的最自然方式。之后，大型软件的系统架构告别了功能分解的垂直分割时代，步入了水平分割时代。这个时代和工业史上的“泰勒时代”有点儿相像——从业人员从事他们最专长的工作。本质而言，大型软件的系统架构从以“问题领域空间”的结构为中心，变成了以“解决方案空间”的结构为中心。

那么下一步呢？我们预言，“否定之否定”的哲学规律将发生作用，迎来横切竖割兼收并蓄的大型软件的系统架构（参见图2）！那么，横切竖割兼收并蓄

的大型软件系统架构如何设计呢？我们认为答案是面向构件。

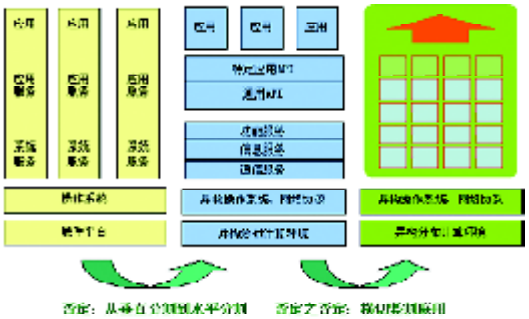


图 2 ：面向构件

面向构件：稳定性、灵活性

此时，软件从业人员依然从事他们最在行的技术，横切时代的优点完全保留了下来。同时，业务构件作为“将不同更小构件编织在一起”的“竖割”的单元，非常符合用户和业务专家的视角，他们可以用业务构件灵活地直接组装所需要的应用。这时，软件业界关于“3.0才开始可用”的“稳定原则”似乎不再是铁律——在你开发产品的1.0版本时，可以大量使用稳定级别在3.0以上的被构件化的软件资产，就可以使1.0版本的产品稳定程度相当高。

面向构件：降低复杂度

面向构件的理想是屏蔽细节、降低复杂性。目前，面向构件方法主要被用来帮助控制大型系统开发的复杂度和风险，在开发的构造和部署阶段中提供一种以架构为中心（Architecture - centric）和以重用为中心（Reuse - centric）的方法。随着时间推移，面向构件方法将演变成所有软件——包括可执行代码、接口规范、架构、业务模型、完整的应用和系统——都成为个体构件，都可以通过跟现成构件组装（Assembling）、适配（Adapting）、“捆绑（Wiring）”在一起完成不同目的。面向构件方法提出了完美的方法来划分业务域，面向构件方法能控制用构件组装大型系统产生的复杂度。这种方法有望将开发的复杂度降低（开发成本随之降低）到八十年代的水平，同时能满足目前的所有需求（参见图3）。

总而言之，通过面向构件的技术，把“黑盒子”“梅花桩”变成了“积木”、“矩阵”，增加了软件的灵活性，降低了软件的复杂度，使企业的业务逻辑随需应变，达到与企业同步成长的目的。

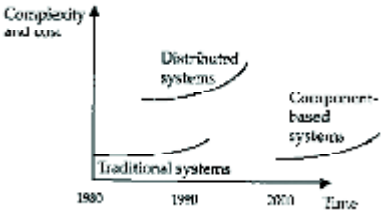


图 3 ：面向构件降低复杂度（本图引用自《Business Component Factory》）

系统容量和性能的成长

“先让系统运行起来，然后让它运行得更快”，不知这算不算业界一条“事实上的标准”。不幸的是，在架构确定之后，想通过调整来获得性能可能为时已晚——至少对于大型软件系统来说，认为“性能只和编码有关”是极为错误的。

从本质上来说，软件性能是计算与计算资源的关系。如图4所示，它采用UML语法描述，“性能”是一个关系类。例如，一个小的桌面应用可能只分配到了PC中单个CPU的10%的计算资源，但由于它计算量较少，因此依然性能极佳；而一个大型互联网应用即使分配了1000台服务器资源，若没有处理好计算与计算资源的关系，其性能仍可能差强人意。

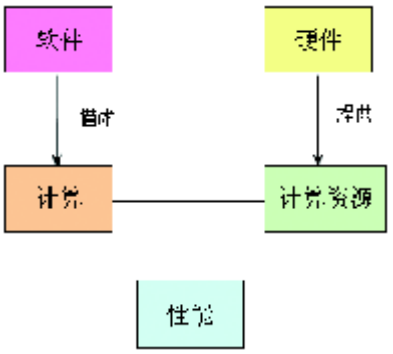


图 4 ：性能的本质——计算与计算资源的关系

因此，软件性能问题，必定首先是架构问题。架构作用于如何组织软件系统、部件的选择、组装部件所需的接口、部件组成更大的子系统时所需明确的行为以及架构风格等方面的重大决定，它从根本上影响着计算和计算资源的关系——系统的性能。在一个大型的应用软件中，建立一个可以优化的架构是设计过程中需要要首先考虑的内容。

数据库性能优化

在为大家普遍关注的大型互联网应用中，因为数据量大、交易量大，在有限的资源面前，性能成为一个不能回避的问题。应用系统往往采用了流行的操作系统、数据库、中间件产品等——这些产品已经在数据和分布处理层上有许多优化措施。在一个大型数据库的查询中，常常有80%的时间都是在数据库端使用的。从而，应用系统的实际性能效果，往往体现在数据库的设计和使用上。然而，关系数据库是一种逻辑描述语言，SQL语句本身的复杂度和计算复杂度之间没有直接关系：我们可以写一个很简单的语句把整个数据库都查询一遍！

将性能策略构件化

面向构件的技术就是通过构件的接口作为软件模块之间的契约，而把实现的细节加以保护。我们可以使用数据和计算的结果，而优化的工作可以在构件内的实现中逐步进行。面向构件的设计原则，也正是重视使系统容量和性能不断成长的稳定的架构。同时，容量和性能提升的知识，也可以以构件为载体，不断的重用。

■ 责任编辑：赵建凯（zjk@csdn.net）